Dipartimento di Informatica, Bioingegneria,
Robotica ed Ingegneria dei Sistemi

Security Analysis of Multi-Factor Authentication Security Protocols

by

Federico Sinigaglia

Theses Series **DIBRIS-TH-2020-03**

DIBRIS, Università di Genova
Via Opera Pia, 13 16145 Genova, Italy http://www.dibris.unige.it/

Università degli Studi di Genova

Dipartimento di Informatica, Bioingegneria,

Robotica ed Ingegneria dei Sistemi

Ph.D. Thesis in Computer Science and Systems Engineering
Computer Science Curriculum

Security Analysis of Multi-Factor Authentication Security Protocols

by

Federico Sinigaglia

May, 2020

Dottorato di Ricerca in Informatica ed Ingegneria dei Sistemi
Indirizzo Informatica
Dipartimento di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi
Università degli Studi di Genova

DIBRIS, Univ. di Genova
Via Opera Pia, 13
I-16145 Genova, Italy
http://www.dibris.unige.it/

Ph.D. Thesis in Computer Science and Systems Engineering
Computer Science Curriculum
(S.S.D. ING-INF/05, INF/01)

Submitted by Federico Sinigaglia
DIBRIS, Univ. di Genova
Security & Trust Research Unit, FBK-Irst, Trento
Date of submission: May 2020

Title: Security Analysis
of Multi-Factor Authentication Security Protocols

Advisors: Roberto Carbone,[1] Gabriele Costa[2]
[1]Security & Trust Research Unit, FBK-Irst, Trento (Italy)
[2]SysMA Unit, IMT School for Advanced Studies, Lucca (Italy)

Ext. Reviewers: Letterio Galletta,[*] Luca Viganò[†]
[†]Department of Informatics, King's College, London (UK)
[*]IMT School for Advanced Studies, Lucca (Italy)

Abstract

Multi-Factor Authentication (MFA) is being increasingly adopted by on- line services in order to achieve an adequate level of security. MFA is based on security protocols, called MFA protocols, that integrate the use of credentials with additional identity proofs, called authentication factors (based on knowledge, possession or inherence). The authentication factors are provided through specific objects, called authenticators (e.g., hardware token). To date, MFA has been widely adopted in the most diverse security-critical application scenarios (e.g., online banking, eHealth). Various solutions have been proposed, leveraging MFA protocols which employ different kinds of authenticators and providing different user experience. When considering various MFA protocols, few questions may arise. How do MFA protocols differ in terms of (i) level of protection, (ii) compliance w.r.t. current regulations and (iii) complexity for the user?

To answer the question concerning the level of protection, traditional verification techniques for security protocols require a formal specification of the protocol under analysis. However, as a matter of fact, several service providers employ ad-hoc MFA protocols and do not disclose their internals. In addition, classical attacker models, such as the Dolev-Yao adversary, hardly apply. Hence, new protocol modeling techniques and new attacker models should be investigated.

Concerning regulations, public and private authorities have introduced directives and guidelines for the design of MFA protocols (e.g., recommendations for online payment services from the European Banking Authority, and the guidelines from NIST about the digital identity management through MFA). In principle, these initiatives aim to guide the design of more secure and usable MFA protocols, but there is no evidence that the existing MFA protocols actually comply with the aforementioned regulations. Thus, a novel methodology is needed to provide such an evidence.

The ease-of-use is a relevant aspect to be considered in the analysis of an MFA protocol. Indeed, the use of multiple authenticators in the execution of an MFA protocol can negatively affect user experience, which can have an impact on its security as well. However, none of the research works managed to measure the usability of a

conspicuous number of MFA protocols design. Hence, a methodology for evaluating the ease-of-use of an MFA protocol should be identified.

In this work, we propose a framework to analyze MFA protocols, which does not rely on the implementation details, being able to assess the (i) level of protection, (ii) compliance w.r.t. current regulations and (iii) complexity for the user.

To this aim, we define a specification language which is compatible with the typical (amount of) information publicly released by service providers on the employed MFA protocols. For what concerns the security analysis, we propose an evaluation of MFA protocols in terms of resistance against a set of attacker models, tailored for the specific case of MFA protocols. For what concerns the regulatory aspects and best practices, we include the possibility to evaluate a protocol in terms of compliance with a customizable set of requirements and best practices. Furthermore, for what concerns the ease-of-use of an MFA protocol, we propose a new metric, called complexity, for evaluating a protocol in terms of efforts that an user is required to perform during its execution.

The aforementioned framework has been then implemented in a working tool, MuFASA, allowing (even non-expert) users to model an MFA protocol and to automatically analyze it.

Finally, the presented framework has been applied on some selected use cases. First, it has been employed in the early stages of the design of a novel MFA protocol, integrated into the Citizens' Clinical Record platform developed in the Trentino region (Italy). Then, it has been used for performing a latitudinary study on online banking services, allowing us to model and analyze more than 150 MFA protocols employed by banks all over the world.

Acknowledgements

Firstly, I wish to show my gratitude to Prof. Alessandro Armando, who believed in me since the beginning of my research activities. Your comments and observations have always been inspiring to me, guiding me towards the best way to proceed in my career. It was an honor to work with you.

I would also like to pay my special regards to my supervisors, Roberto Carbone and Gabriele Costa. I am very grateful for your support, for your patience, for your care and for all the discussions that helped me in doing my best in every aspect of my research.

I would also like to thank Prof. Giorgio Delzanno, coordinator of my Ph.D. Programme, the technical committees and the reviewers.

Special thanks are also due to Silvio Ranise, head of the Security & Trust Unit of Fondazione Bruno Kessler: thanks for the unconditional support you have given me during these years.

I obviously extend my thanks to the whole Security & Trust Unit, where I have spent most of my time during my research activity. I would like to thank each one of you, newer, older and past members: it has been a pleasure to work with you.

I also want to express my gratitude to Nicola Zannone, that helped me in reaching one of the best achievements of my research work. Thanks, Nicola, for your support and patience. It has been an honor to work with you.

Finally, I would like to thank my girlfriend, my family, the Family (i.e., my long-time friends), all my friends and all the people that supported me during these years, for your love, patience and encouragement. I could not have made it without you.

Table of Contents

Glossary

Digital Identity A digital identity (as defined in [NIS17]) is as a set of attributes that uniquely describe a user in a specific context (e.g., a payment service).

Authentication The authentication is a process aimed at verifying users' identity, thus constituting a prerequisite to allowing access to resources (as defined in [NIS17]). The verification of a user's digital identity is performed through a so-called *authentication protocol.*

Authentication Protocol An authentication protocol, as defined in [NIS17], is a sequence of actions that allow the *digital authentication* of a user by verifying the possession and control of specific categories of credentials called *authentication factors* (by NIST in [NIS17]) or *authentication elements* (by EBA in [Eur13a, Eur13b, Eur15, Eur17]).

Authentication Factor According to both NIST and EBA, an authentication factor (or authentication element) can be of three different types: (i) something the user knows (*knowledge factors*); (ii) something the user possesses (*ownership factors*); or (iii) something the user is (*inherence factors*). When an authentication protocol leverages more than one authentication factor, it is referred to as MFA protocol.

Authenticator An authenticator, as defined in [NIS17], is an object attesting one or more authentication factors. An authenticator is "something the user possesses and controls (typically a cryptographic module or password) that is used to authenticate the user's identity". An authenticator can attest more than one authentication factor in which case it is referred to as *multi-factor authenticator*.

Authenticator Output The authenticator output is a value that every authenticator can generate on demand ([NIS17]). The ability to generate valid authenticator outputs proves that the user possesses and controls the authenticator (and thus the corresponding authentication factors). Nevertheless, the relationship between an authenticator and its output depends on the nature of the authenticator itself. In the case of a knowledge factor, for instance, an authenticator and the corresponding output are the same entity (e.g., the password or the secret code itself).

Authentication Code According to EBA, an authentication code is a unique code generated by a cryptography-based authenticator and *dynamically linked* to a specific *remote payment transaction* (as presented in [Eur17]). We consider the authentication code as a specific authenticator output.

Dynamic Linking With Dynamic Linking, we indicate a set of security measures to be adopted by bank s.t. (i) the user is made aware of the details of the operation she is going to authorize; (ii) the authentication code generated is specific details of the operation; (iii) the authentication code accepted by the payment service provider corresponds to the original details of the operation chosen by the user; (iv) any change to the details of the operation results in the invalidation of the authentication code generated.

Endpoint The endpoint, as defined in [NIS17], is the platform or device, i.e., a web browser or a mobile phone, from which the user starts the MFA protocol.

Verifier A verifier is "an entity that *verifies* the user's identity by verifying the user's possession and control of the authenticators" [NIS17]. The type of verifier depends on the nature of the corresponding authenticator. In the case of a memorized secret (e.g., a password), for instance, the verifier has "only" to check if the provided secret is correct. In the case of an OTP device, instead, the verifier has to generate an expected OTP and compare it with the received one in order to validate it. In this work, the verifier coincides with the service provider server.

Enrollment The Enrollment (defined in [NIS17]) is the phase in which the user is registered to the service. It encompasses a preliminary process of user identification, called *identity proofing*. Through this process, a service provider collects, validates and verifies information about an individual.

Binding The binding (according to [NIS17]) is the procedure through which the identity proofs (i.e., the authenticators and the authentication factor they attest) are linked to the user identity. This procedure is also called *delivery of credentials, authentication devices and software* by EBA in [Eur15, Eur17]). In this work, we further detail the binding phase in three steps: *request*, *delivery* and *activation*. During the request step, the user informs the bank that she wants to activate a certain AF. The second step concerns the delivery of authenticators to the user. The activation step aims to guarantee that AFs are properly delivered. Binding can be executed by a human operator or remotely, e.g., over the Internet or via registered mail, potentially leveraging an MFA protocol employing previously bound authenticators. Note that additional binding operations can be performed in a separate occasion, i.e., whenever a user wants to associate a new authenticator to her identity.

Exemptions The exemptions are defined in [Eur15, Eur17] as a set of situations, identified a slow-risk operations, in which MFA may not be employed. In the online banking context, these situations can be: (i) checking the account balance, (ii) paying a trusted beneficiary, (iii) executing a recurring transaction and (iv) executing a bank transfer between user's own accounts.

MFA implementation With MFA implementation, we refer to the design choices taken by an online service in terms of adoption of MFA protocols, authenticators, enrollment and binding procedures

Chapter 1

Introduction

1.1 Context and Motivations

In recent years, the usage of online services has considerably increased. Nowadays, user can manage their banking accounts, control their personal health records or access to local administrations services just by connecting to a web platform. The availability of the most diverse services have grown, allowing people to perform online what they were required to execute in person. However, although online services provide evident benefits to both service provider themselves and customers, they introduce new security and privacy issues. Data leakages or data thefts are a growing concern, as they can harm users in different ways. Therefore, protecting sensitive data has become of paramount importance.

A fundamental security measure for the protection of online resources is the employment of reliable (digital) authentication mechanisms, i.e., procedures that verify the digital identity of users and check their legitimacy. In this context, users have to exhibit an identity proof that can only be provided by the users themselves, thus deterring attackers from breaching their online resources. The most common identity proof consists of user credentials, i.e., username and password. However, due to the naive behavior of users (e.g., employing the same guessable passwords on several online services) and the increasing strength of malicious agents that currently operate over the internet, they are often considered insufficient to achieve an adequate level of security and their use exposes users to several threats.

To tackle this problem, Multi-Factor Authentication (MFA) is being increasingly adopted by online services. MFA is based on security protocols, called MFA protocols, that integrate the use of credentials with additional identity proofs, called *authentication factors* [NIS17]. Authentication factors are based on either *knowledge*, *possession* or *inherence*. During the execution of an MFA protocol, authentication factors are provided through specific objects, called *authenticators* (see [NIS17]). Therefore, an MFA is designed to force a malicious user to adopt various techniques

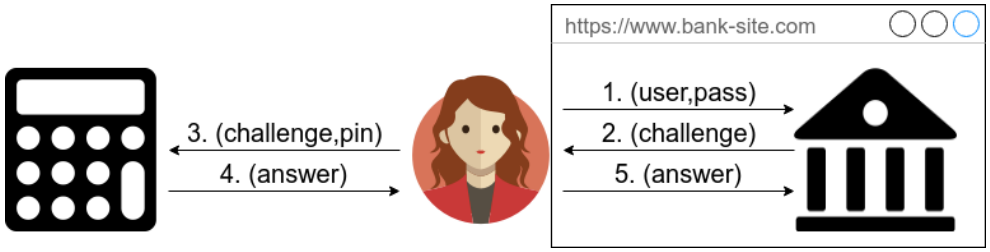


Figure 1.1: One among the MFA protocols adopted by Nordea - Nordea1.

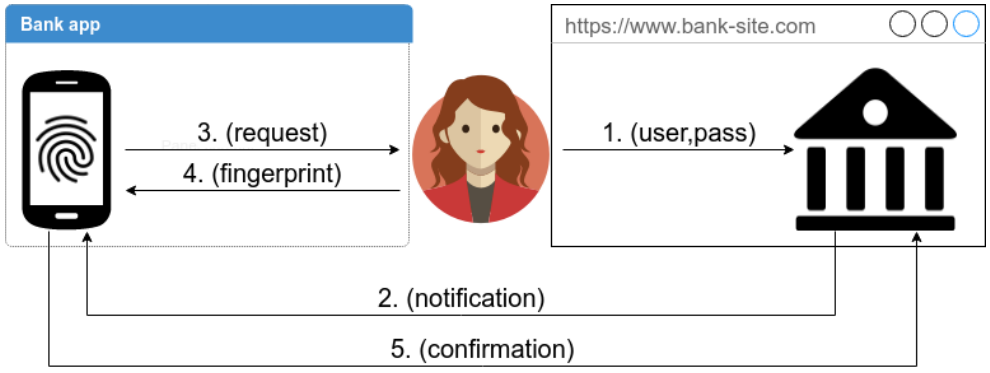


Figure 1.2: Another one among the MFA protocols adopted by Nordea - Nordea2.

for compromising the authenticators and authenticating instead of the user.

To date, MFA has been widely adopted in the most diverse application scenarios. The majority of online banking services, eHealth platforms and identity providers implements MFA, proposing extremely various solutions. Indeed, for authenticating to online services, users are commonly offered multiple choices of MFA protocols employing different kind of authenticators and providing different user experience. To better highlight this situation, we propose the following motivating example.

Motivating Example. *Let us consider the online banking services of Nordea*[1]*, a major, international bank based in Sweden. Its customers can opt, among others, for two MFA protocols for authenticating, leveraging a physical device*[2] *and a mobile application*[3]*. We schematically depict them in Figures 1.1 and 1.2.*

In the first MFA protocol (Nordea1, from now on) the user is in possession of a device to authenticate. Initially, she connects to the website of the bank and she logs in with her credentials - a memorized secret*, attesting a* knowledge factor *(1). Then (2) she is prompted with a challenge, i.e., a code that only the device can process, and (3) she submits the challenge to the device*

[1] https://www.nordea.se/

[2] https://www.nordea.se/Images/154-21252/quickguide-cardreader.PDF

[3] https://www.nordea.se/Images/154-300029/Broschyr_skaffa_Mobilt_BankID.pdf

(together with the card and the pin code that unlocks it) - a multi-factor authenticator*, attesting both an* ownership *and a* knowledge *factor. Finally, the device returns an answer code (4) that the user copies in her browser (5).*

The second protocol (Nordea2, from now on) is slightly different. Again, the process starts with the user authenticating to the website (1), hence providing a memorized secret*. However, in this case the remote service sends a notification (2) to a mobile application running on the user's smartphone (an* Out-of-band authenticator*). The application displays the authentication request (3) to the user that, to confirm, touches the fingerprint reader of the phone, providing and* inherence *factor (4). The protocol terminates (5) with the mobile app sending a confirmation code to the bank server.*

When considering these two MFA protocols, few questions arise. In particular, do the two protocols differ in terms of (i) level of protection, (ii) compliance w.r.t. current regulations and (iii) complexity for the user?

For what concerns the level of protection, a traditional approach is the formal verification of the security protocols. There is a rich literature on security protocols that have been either formally validated or proved flawed through this kind of techniques (e.g., [Low96, APS14, BV11, BM11, CGDW09])). These verification techniques require a formal specification of the protocol under analysis. Some authors followed this approach also for the MFA protocols (as [ACZ13, DeF11, Hag07, JCL$^+$17, JJIL16]. However, this can be done only for an MFA protocol when its implementation or design are given. In many real cases, this is unlikely to happen. As a matter of fact, several service providers employ ad-hoc MFA protocols and do not disclose their internals. Therefore, modeling their MFA protocols with traditional techniques is often unfeasible. As a further consequence, even classical attacker models, such as the Dolev-Yao [DY81] adversary, hardly apply. Hence, new protocol modeling techniques and new attacker models should be investigated.

In parallel with the aforementioned initiatives on MFA protocol analysis, public and private authorities (belonging to various application scenarios) have introduced regulations, directives and guidelines for the design of MFA protocols. For instance, the European Banking Authority (EBA) acknowledged the importance of MFA in the online banking context and, starting from 2013, issued directives and recommendations for online payment services [Eur13a, Eur13b]. The most recent directive is [Eur15] and further related regulatory standards have been also published (e.g., [Eur17]). Similarly, the National Institute of Standards and Technology (NIST) [NIS17] and Payment Card Industry (PCI) [PCI17] have proposed a set of guidelines concerning the digital identity management through MFA. Analogous initiatives are also carried out by private companies, which have started releasing their own guidelines [Cen16, Gem15, Pin09]. In principle, these initiatives aim to guide the design of more secure and usable MFA protocols. However, there is no evidence that the existing MFA protocols actually comply with the aforementioned regulations and best practices. Thus, a novel methodology is needed to provide such an evidence.

The ease-of-use is another relevant aspect to be considered in the analysis of an MFA protocol. As shown in [CDFN13, KPCS15, WDRJ10], the use of multiple authenticators in the execution of an MFA protocol can negatively affect user experience, which can have an impact on its security. However, the majority of research works on the topic (as [DDC18, WDCJ09, Alt16]) usually consists in usability tests performed on the field, involving small groups of people and surveying their perception on the usage of given sets of MFA protocols. Given the number of existing MFA protocols and the objective difficulty in presenting to subjects a large set of protocols to test, none of the authors managed to measure the usability of a conspicuous number of MFA protocols design. Hence, a methodology for evaluating the ease-of-use of an MFA protocol should be identified.

In this work, we propose a framework to analyze MFA protocols relying on neither implementation details nor behavioral specifications. To this aim, we define a specification language which is compatible with the typical (amount of) information that is publicly released by service providers on the employed MFA protocols. In practice, we model a protocol in terms of the operations that a user is required to perform during an authentication session, i.e., in a setting that is the same of our motivating example. In particular, our language focuses on the concept of authenticator as the basic building block of any MFA protocol. The semantics of our language is given in *Security Protocol Specification* (SPS) language [AMV15].

For what concerns the security analysis, we propose an evaluation of MFA protocols in terms of resistance against a set of attacker models. These attackers have been tailored for the specific case of MFA protocols, being modeled following dedicated literature (as [JFR17, Ayy17, LM11]) and NIST definitions [NIS17]. For what concerns the regulatory aspects and best practices, we include the possibility to evaluate a protocol in terms of compliance with a customizable set of requirements and best practices. For presenting this feature, we extracted and encoded a set of requirements (deriving from EBA regulations) and a set of best practices (derived from [NIS17, PCI16a] and others) and use them for evaluating a protocol. Furthermore, for what concerns the ease-of-use of an MFA protocol, we propose a new metric, called *complexity*, for evaluating a protocol in terms of efforts that a user is required to perform during its execution.

The aforementioned framework has been then implemented in a working tool, MuFASA. By showing a user-friendly interface, MuFASA allows (even non-expert) users to model an MFA protocol and to automatically analyze it, giving a report presenting the performed evaluations and the obtained results.

Finally, the presented framework has been applied on some selected use cases. First, it has been employed in the early stages of a new MFA protocol design. Then, it has been used for performing a latitudinary study on online banking services, allowing us to model and analyze more than 150 MFA protocols employed by banks all over the world.

1.2 Contributions

The contributions of this thesis cover both the specification and the analysis of an MFA protocol. More in detail, the contributions are the following.

1. The definition a new specification language for modeling MFA protocols and their preliminary phases (Chapter 3). Such language allows for specifying the main features of an MFA protocol design, abstracting the details of actual protocol implementations. Moreover, it is quite simple to employ, since the specification is based on client-side actions and communications that can be easily identified even by non-expert users. The semantics of our language relies on SPS: a modeling language that is specifically designed to represent security protocol with a proper abstraction level.

2. The definition of a framework for the analysis of an MFA protocol (Chapter 4). The framework features a set of threat models for the specific context of MFA. Moreover, it allows for evaluating a protocol in terms of compliance with a set of requirements and best practices that can be customized on the application scenario. Furthermore, it leverages a newly defined metric, called *complexity*, for evaluating the ease-of-use of a protocol basing on objective measurements.

3. The implementation of the presented framework in a working prototype, the MuFASA (Multi-Factor Authentication Specification and Analysis) tool, allowing for specifying and automatically analyzing MFA protocols (Chapter 5). Leveraging such tool, an extensive application of the proposed methodology has been performed, obtaining encouraging experimental results on the analysis of MFA protocols employed by different banks around the world (Chapter 6).

1.3 Structure of the Thesis

The thesis is structured as follows.

Chapter 2 - Background

In this chapter, we introduce the reader to the basic notions and definitions on which this thesis work is based. To this aim, the first section presents the definition of *Digital Identity*. Secondly, we present the basic notions regarding *Authentication protocols* and our basic assumptions. Furthermore, a section concerning MFA is presented. In particular, this section presents the definitions of key concepts related to MFA proposed both by NIST and an relevant institution in the online banking scenario: the European Banking Authority. The given definitions are compared and merged in a set of notions that will be used throughout this thesis.

Chapter 3 - Specification

In this chapter we introduce the reader to our language for specifying MFA protocols. First, we discuss the key features of our language and the motivations behind its development. Then, we present the language itself, defining both its syntax and semantics, in terms of translation to SPS [AMV15], a variant of Alice & Bob language. For each part of the translation, an example involving the Nordea1 protocol is provided. Furthermore, a section related to the compliance of our language with NIST definitions is presented. Finally, the language for specifying the preliminary phases of an MFA protocol is defined.

Chapter 4 - Analysis Framework

In this chapter, we present our analysis framework. In particular, we firstly define the set of attacker models (and their application conditions) that can be considered for the protocol analysis. Furthermore, we describe the possibility of including evaluation criteria related to best practices and security requirements deriving from the scenario in which the protocol will be executed. Finally, we introduce the *complexity*, a metric for evaluating the efforts required by users for executing the protocol.

Chapter 5 - Framework Implementation

In this chapter, we present MuFASA, a tool implementing the our specification language and the analysis framework. At first, we introduce the user to the features and the goals of the protocol. Then, we present the tool architecture and the implementation details. Finally a demonstration of the execution flow is given.

Chapter 6 - Results

In this chapter, we present the experimental results that we obtained by applying our methodology on different use cases. At first, we discuss how our methodology has been adopted during the design of a new MFA protocol for eHealth services. Then, we present a latitudinary study on a dataset of banks that implement various MFA protocols for protecting their online services. Finally, we discuss potential threats to validity of our findings.

Chapter 7 - Related Work

In this chapter, we present the related work. In particular, we compare our work to existing publications, in different aspects of protocol analysis.

Chapter 8 - Conclusions

Some of the material in this manuscript is based on the following journal and papers published at international conferences:

- Sinigaglia F., Carbone R., Costa G. and Zannone N., *A Survey on Multi-Factor Authentication for Online Banking in the Wild*, In Computers & Security, 2020, https://doi.org/10.1016/j.cose.2020.101745.
- Sinigaglia F., Carbone R., Costa G., and Ranise S., *MuFASA: A Tool for High-level Specification and Analysis of Multi-factor Authentication Protocols*, In: Saracino A., Mori P. (eds) Emerging Technologies for Authorization and Authentication. ETAA 2019. Lecture Notes in Computer Science, vol 11967, Springer, https://doi.org/10.1007/978-3-030-39749-4_9.
- Sinigaglia F., Carbone R. and Costa G., *Strong Authentication for e-Banking: A Survey on European Regulations and Implementations*, In Proceedings of the 14th International Joint Conference on e-Business and Telecommunications - Volume 4: SECRYPT, (ICETE 2017) ISBN 978-989-758-259-2, pages 480-485, https://doi.org/10.5220/0006438504800485.

Chapter 2

Background

In this preliminary chapter we introduce the reader to the basic concepts and definitions on which this work is based.

2.1 Digital Identity

The main topic of this thesis is the so-called authentication. With the term *authentication*, [Low97] indicates the process through which "one agent should become sure of the identity of the other". However, this definition implicitly relies on another one, i.e. the definition of *(digital) identity*.

Intuitively, the digital identity can be considered as the "online representation" of a user. A precise and unique definition is still on debate: indeed, depending on the context, the user may be represented through different ways (e.g., with an email, a username, a software, etc.). However, by taking the definition given by NIST it in [NIS17], the digital identity can be generically defined as "the unique representation of a subject engaged in an online transaction".

Further refinements to this generic definition derive from the specific context in which a user acts. This fact is also remarked in [NIS17], in which NIST states that "a digital identity is always unique in the context of a digital service, but does not necessarily need to uniquely identify the subject in all contexts". In other words, the digital identity of a user could be based on her attributes, features or just an email, depending on the specific needs of the application scenario.

Nevertheless, regardless of the application scenario, the creation of a digital identity must be consequent to a process called *identity proofing*. This process consists in establishing that a user is who she claims to be. Once the process has been successfully completed, the user's identity for the specific context (or service) can be established.

An authentication process between the user and the server will hence aim at letting the server identify the user by recognizing her digital identity. To this aim, it is necessary to establish a set of elements, called *identity proofs*, on which to perform the identification. These identity proofs are associated to the specific digital identity and can be of various nature (depending on the context). The most classic example of identity proof is a set of credentials (i.e., username and password). By providing the required identity proofs, the user can convince the server of her identity, hence being authenticated.

2.2 Authentication Protocols

As the number of personal and sensitive data that is managed by online services grows, defining mechanisms for transmitting and access data in a secure way is of increasing importance. Security protocols are exchanges of messages between two or more agents that try to achieve one (or more) security goal(s), running in potentially hostile environments. Among the possible security goals, including the *establishment of a shared key* or performing a *confidential transmission of a datum*, the goal of *authenticating one agent to another* is the one of our interest.

Achieving the authentication means that, at the end of the protocol, one agent should become sure of the identity of the other. Trivially, an *Authentication protocol* is a security protocol that is designed to assure an agent as to the identity of the other agent with whom is running the protocol is called.

2.2.1 Cryptographic Primitives

Authentication protocols (security protocols in general) usually rely on some form of cryptographic operations to achieve their security goals.

In particular, cryptographic operations can be used to guarantee the confidentiality of a message, to certify the identity of the sender, to prevent repudiation, etc.

In this work, we will consider authentication protocols relying on both *symmetric* and *asymmetric* cryptography.

Symmetric Key Symmetric encryption is a type of encryption where only one secret key is used to both encrypt and decrypt a message. In other words, anyone who possesses the key can encrypt a message that is confidential to others that possess the same key. On the other hand, anyone who possesses the key can decrypt and read any message that has been encrypted with the same key.

Asymmetric Key The Asymmetric encryption is based on a pair of keys: a public key and a private key. While the first is made publicly available, the second must be kept secret and must be properly protected by its owner. These two keys are one the dual of the other. This kind of encryption is commonly used for two operations:

Public key encryption. In this case, the public key is used to encrypt the message, while private key is used to decrypt. In this case, the encrypted message can be decrypted only by theowner of the private key.

Digital Signature. In this case, the message is signed with the private key. Anyone can read the message, leveraging the public key, implicitly authenticating the identity of the sender.

As usual in the protocol analysis, we assume perfect cryptography. This means that attackers can encrypt or decrypt messages only if they possess the correct key.

2.2.2 Channel Properties

In this work, following what presented in [Pro08] we consider channels as communication mediums through which a participant A can transfer data to another participant B. In particular, we assume that each channel be of several kinds, depending on the security properties it guarantees:

Confidential We assume a channel to be confidential if A can be sure that any message sent over this channel can only be read by B. In other words, any intruder cannot learn the contents sent on this channel. Confidential channels do not protect the message transmission against intercepting messages and replacing them with different ones, spoofing (the intruder may introduce messages on the channel in the name of other parties), disruption (the message does not arrive at the intended receiver), replay or re-ordering (messages may not arrive at the recipient in the order they were sent).

Authentic We assume a channel to be authentic if B can rely that any datum he receives via this channel indeed comes from A. Moreover, we assume that this kind of channels guarantees the integrity of the transmitted message. Furthermore, in all channel models, for channels that are at least authentic, B can be sure that the message was meant for him. The authentic channel does not protect message transmission against eavesdropping, disruption, replay or re-ordering.

Secure A channel is defined *secure* if it is both confidential and authentic.

2.3 Multi-Factor Authentication

In literature, it has been very difficult to obtain a precise definition of MFA. Indeed, we observed a lack of a common and consistent terminology in the field: several unofficial definitions, based on different key features and requirements, have been written depending on the context (or the environment) in which MFA is implemented. Among the others, we have identified two main authoritative bodies, namely the National Institute of Standards and Technology (NIST) [NIS17] and the European Banking Authority [Eur13a, Eur13b, Eur15, Eur17]. These authorities target different aspects of MFA, i.e., the application of MFA for digital authentication management and for online payment services, respectively. In the following sections, we present the key concepts characterizing the two sources. Finally, we present the common terminology that will be used throughout the thesis, obtained after revising and aligning the concepts identified by the two sources.

2.3.1 Regulatory Definitions - EBA

In order to overcome the lack of guidelines and definitions, lawmakers and institutions started to publish a few documents to steer the design of this new kind of authentication. However, lawmakers belonging to different application scenarios provided different interpretations and specifications. On top of that, these "early" documents did not provide sufficient information to obtain a homogeneous landscape in the protocols design.

Among different application scenarios, online banking is one of the first ones in which the importance of having stronger authentication mechanisms (if compared to "normal" ones) has been acknowledged. Starting from 2013, several documents [Eur13a, Eur13b] and directives [Eur15] have been published, progressively refining and enriching the key definitions characterizing the MFA until 2017, when various regulatory standards (e.g., [Eur17]) have been released. Therefore, given this progressive evolution and precision of these definitions, we decided to take them as a reference in our work.

In the following sections, the definitions provided by EBA in [Eur13a, Eur13b, Eur15, Eur17] are presented, grouped according to their scope: *authentication*, *identity proofs* and *preliminary phases*.

2.3.1.1 Authentication

The MFA is defined by EBA in [Eur13a], with the name of "*Strong Customer Authentication*". In this context, the Strong Customer Authentication has been firstly defined in [Eur13a] as: "a procedure based on the use of two or more of the following elements – categorised as knowledge, ownership and inherence: i) something only the user knows, e.g., static password, code, personal

identification number; ii) something only the user possesses, e.g., token, smart card, mobile phone; iii) something the user is, e.g., biometric characteristic, such as a fingerprint. In addition, the selected elements must be mutually independent, i.e. the breach of one does not compromise the other(s)".

2.3.1.2 Identity proofs

It is worth noting that the definition of Strong Customer Authentication is based on the concept of *authentication elements*. These elements can be of three types (knowledge, ownership and inherence) and must be used to verify the user's identity.

In [Eur15] and [Eur17], EBA introduced the concept of *authentication code*. According to EBA, the authentication code must be the result of the execution of the Strong Customer Authentication. This code must be only accepted once by the service provider and must be resistant against the risk of being forged in its entirety or by disclosure of any of the elements upon which the code was generated.

On the top of that, EBA defines the concept of *dynamic linking*. The dynamic linking is supposed to bind the Strong Customer Authentication to a specific operation. To this aim, the following measures must be implemented

- the user is made aware of the amount of the payment transaction and of the payee;
- the authentication code generated is specific to the amount of the payment transaction and the payee agreed to by the payer when initiating the transaction;
- the authentication code accepted by the payment service provider corresponds to the original specific amount of the payment transaction and to the identity of the payee agreed to by the payer;
- any change to the amount or the payee results in the invalidation of the authentication code generated.

2.3.1.3 Preliminary phases

Besides the requirements listed so far, EBA defines further specifications regarding the preliminary phases, i.e., those procedures that have to be completed for allowing a user to execute an MFA protocol. In particular, [Eur13a] includes some basic requirements for the registration of the user to the online service and the establishment of the authentication elements. These phases, called *enrollment* and *provisioning* by EBA, are crucial for the security of the Strong Customer Authentication. Indeed, if the authentication elements are compromised during the preliminary

phases by a malicious agent, he can easily authenticate instead of the user. About these two phases, [Eur13a] says: "PSPs should ensure that customer enrolment for and the initial provision of the authentication tools required to use the internet payment service and/or the delivery of payment related software to customers is carried out in a secure manner".

This definition is refined in [Eur17], in the paragraph "delivery of credentials, authentication devices and software".

2.3.2 General Purpose Definitions - NIST

In parallel with regulatory aspects belonging to specific use cases, other institutions have started to publish guidelines and documentations on MFA that can be adopted in various application scenarios.

Among those, NIST has recently published a document concerning the management of a Digital Identity [NIS17]. Given both the relevance of the institution and the completeness of the published guidelines, we decided to take [NIS17] as a reference for our work. In the following, a subset of the definitions presented in [NIS17], grouped according to their scope, are reported.

2.3.2.1 Authentication

The definition of authentication provided by NIST is based on the concept of user's digital identity, which is defined as "a set of attributes that uniquely describe a user in a specific context". According to [NIS17], the verification of a user's digital identity is performed through a so-called authentication protocol, that allows for the *digital authentication* of the user.

An *authentication protocol* is "a sequence of actions that allow the digital authentication of a user by verifying the possession and control of specific categories of credentials called *authentication factors*".

2.3.2.2 Identity proofs

According to NIST, an *authentication factor* can be of three different types: (i) something the user knows (knowledge factors); (ii) something the user possesses (ownership factors); or (iii) something the user is (inherence factors). When an authentication protocol leverages more than one authentication factor, it is referred to as *MFA protocol*.

In addition to the Authentication Factors, the NIST defines the concept of *authenticator*. An authenticator is "something the user possesses and controls (typically a cryptographic module or password) that is used to authenticate the user's identity". An authenticator is an object that

Table 2.1: Used Definitions.

Definition	NIST	EBA
Digital Identity	✔	✔ (User identity)
Authentication	✔	✔
Authentication Protocol	✔	✔
Authentication Factor	✔	✔ (Authentication Element)
Authenticator	✔	✖
Authenticator Output	✔	✖
Authentication Code	✖	✔
Dynamic Linking	✖	✔
Endpoint	✔	✖
Verifier	✔	✖
Enrollment	✔	✔ (Registration to the service)
Binding	✔	✔ (Delivery of credentials, authentication devices and software)
Exemptions	✖	✔
MFA Implementation	–	–

attests one or more authentication factors. Every authenticator can generate an output value on demand, called *authenticator output*. The ability to generate valid authenticator outputs proves that the user possesses and controls the authenticator (and thus the corresponding authentication factors). In [NIS17], multiple types of authenticators are defined, along with their key features and references to real-world examples.

For what concerns the platform from which the authentication is started by the user, the NIST refers to it as the *endpoint*.

2.3.2.3 Preliminary phases

NIST identifies two preliminary phases for the proper execution of the MFA: *enrollment* and *binding*.

The *enrollment*, i.e., the registration phase, encompasses a preliminary process of user identification, called *identity proofing*. Through this process, a service provider collects, validates and verifies information about an individual. Once this process is performed, the service provider is able to recognize the identity of the individual with an adequate level of assurance.

The so-called *binding* phase, instead, refers to the establishment of an association between a specific authenticator and a subscriber's account, enabling the authenticator to be used — possibly in conjunction with other authenticators — to authenticate for that account. The binding procedure can be executed both during or after the enrollment phase.

2.3.3 Employed Definitions

The glossary of this thesis contains a common terminology that will be used throughout this work. Such terminology has been obtained after revising and aligning the concepts identified by the two sources. A brief recap of the sources is presented in Table 2.1.

Chapter 3

MFA Specification

In this chapter we present our methodology for the specification of MFA protocols and preliminary phases.

The following section presents an overview of the approach. Then, the language for modeling MFA protocols is detailed, in terms of both syntax and semantics. Moreover, compliance of the presented language w.r.t. NIST definitions (listed in [NIS17]) is evaluated. Finally, the language for the specification of preliminary phases is proposed.

3.1 Overview

We developed a specification language using the *authenticators* as building blocks for specifying a protocol: SLaMP - **S**pecification **La**nguage for **M**ulti-Factor Authentication **P**rotocols.

Several reasons support our design choice. First of all, authenticators are the key objects in an MFA protocol execution, since they attest the authentication factors that allow the user for being authenticated. Furthermore, a finite set of authenticator types, that can be used in MFA protocols, is defined by NIST in [NIS17]: this means that common design patterns, features and security considerations can be found in MFA protocols using the same authenticators. Finally, various distinguishing features of authenticators can be easily described (even by non-expert users) even without having details on actual protocol implementations. Hence, in SLaMP, we consider an MFA protocol as *a sequence of authenticators*, where each authenticator subsumes a set of communications, interactions with other entities and security assumptions that characterize the protocol.

The SLaMP language subsumes a set of operations and communications that are performed during the protocol execution. The semantic of SLaMP is given in the *Security Protocol Spec-*

ification language[AMV15] (SPS, from now on), a variant of the *Alice & Bob* language with the addition of few information regarding the messages schemas, the transmitted data and the communication channels.

In order to assess the proposed language, the compliance of SLaMP with respect to NIST classification is performed.

Finally, for what concern the specification of preliminary phases, we decided to introduce a finite set of symbols indicating the different ways to perform an identity proofing and binding procedure. Indeed, following what presented in [NIS17] and observing real-world solutions, we realized that few patterns in the executions of such procedures can be identified. Therefore, a few symbols indicating these patterns are employed.

3.2 SLaMP protocol specification

In this section, SLaMP is presented. The idea is to model each MFA protocol in terms of the authenticators it involves.

Therefore, we specify an MFA protocol P as a finite sequence of authenticators $(A_1; \ldots; A_n)$, using ";" to separate the elements of the sequence.

In our framework, an authenticator is uniquely characterized by four main features, i.e., its (i) type, (ii) I/O channels, (iii) data and (iv) the authentication factors it attests. The general definition of an authenticator is

$$\delta \gg_{\gamma} \tau^{(?)}[F] \gg_{\gamma'} \delta' \tag{3.1}$$

where δ, δ' are data items (discussed in Section 3.2.3), γ, γ' are channels (discussed in Section 3.2.2), τ is an authenticator type (discussed in Section 3.2.1) and F is the set of authentication factors that the authenticator attests. In addition, an authenticator type may be optionally labeled with **?** [1] to indicate that, for each operation, it requires the review and confirmation of the user (discussed in Section 3.2.4).

We define Ω as the set of all authenticators that can be specified in SLaMP.

In Table 3.1, a brief summary of the notation employed in our modeling language is reported. In the following sections, we detail how the various aspects characterizing an authenticator are modeled.

[1] We use () to denote optional terms.

Table 3.1: Notation for specifying authenticators in SLaMP.

Authenticator type	**Data items**		**Data channels**	
Memorized secret	ε	No input	$\gg_h$	Manual copy
Look-up secret	`opid`	Operation identifier	$\gg_i$	Inter-process communication
Hardware authenticator	`otp`	One-time password	$\gg_o$	Optical code scan
Software authenticator			$\gg_m$	Mobile telephony network

Authentication factors	**Additional notations and restrictions**
K Knowledge Factor	Out-of-band hardware authenticator
O Ownership Factor	Out-of-band software authenticator
I Inherence Factor	$\tau^?$ User confirmation required

3.2.1 Authenticator Types

The type of an authenticator is defined by its physical and behavioral features. We distinguish among three categories of authenticators, i.e., *secrets*, *hardware* and *software*. Below we describe them in details.

3.2.1.1 Secrets

A secret is the simplest form of authenticator. Basically, it consists of some information, e.g., a secret number, that the user exhibits at a certain point of the protocol. We distinguish between *memorized secrets* () and *look-up secrets* (). The main difference between these two types is that a memorized secret represents a pure knowledge factor, e.g., a secret pin number, while a look-up secret is stored on some physical support such as a device or a piece of paper. For instance, look-up secrets include matrices where each cell contains an access code.

3.2.1.2 Hardware

Hardware authenticators () are devices that carry out some computation, e.g., cryptographic operations. Hardware authenticators can either work in isolation (i.e., only interacting with the user) or connect to some other device (e.g., through a USB cable). Nevertheless, they consist of dedicated hardware and they cannot carry out any task other than that specified by their embedded program.

The AFs of a hardware authenticator can vary significantly. For instance, a card reader may also require a secret pin, i.e., it attests both an ownership (the smart card) and a knowledge (the pin) factor. Notice that the pin of the card reader is not considered a memorized secret (see above) if it cannot be used as a stand-alone authenticator, e.g., when its purpose is only to protect the

hardware from unauthorized users. We denote with 🖩[F] a hardware authenticator that attests the AFs $F \subseteq \{\text{O, K, I}\}$. For instance, (the type of) the card reader mentioned above is 🖩[O,K].

3.2.1.3 Software

Software authenticators (🔐) are the counterpart of hardware authenticators. Their distinguishing feature is that they consist of programs that run on some general-purpose computing platform, e.g., a tablet or a smartphone. As for the hardware authenticators, also software authenticators can attest different AFs. Hence, we apply the same notation introduced above. For instance, 🔐[O] stands for (the type of) a software authenticator that attests an ownership factor.

Example 1. *Both Nordea1 and Nordea2 start with the user entering her credentials. This corresponds to a memorized secret* 🔑*. Instead, the two protocols differ on the second authenticator. In the first case, there is a hardware authenticator that is* owned *by the user and is activated by a pin that she only* knows*. In our language this corresponds to* 🖩*[O,K]. In the second protocol, the second authenticator is a software running on the smartphone of the user that the user activates through her fingerprint. In symbols this amounts to* 🔐*[O,I].* □

3.2.2 Data Channels

MFA protocols often rely on more than one communication channel. There are several types of channels that are commonly adopted and we list them below.

h *Human beings* are part of the MFA protocols. Their role is often to provide the authenticators with the proper input and collect their output. For this reason we model them as communication channels.

o Sometimes an authenticator acquires its input through some *optical* scan interface, e.g., a bar code or QR code reader.

n The *network* is the primary communication channel for most internet protocols. The network channel includes all IP-based communications independently from the link medium (e.g, WiFi or 4G).

i Sometimes two processes directly communicate through some *inter-process* channel. For instance, this is very common on the modern smartphones where two apps can directly share a piece of data.

m The *mobile telephony network* is also commonly adopted to send authentication codes through an SMS of a phone call.

Example 2. *Let us consider again the two MFA protocols presented in Section 1.1. In the first protocol every piece of information is transmitted by the user, i.e., user, password, challenge, pin and answer. Thus, all these operations occur on the h channel. In the second protocol, the user only inserts her credentials. All the other communications pass through the network (n channel).* □

3.2.3 Data Items

Data items represent the information that an authenticator receives and generates. Although the internal structure of the communications may be obscure, e.g., because messages are encrypted, the content of a data item is commonly well-understood. For instance, a hardware authenticator may return a number to be submitted on a website in order to authorize an operation. Such a number can be generated through various algorithms (e.g., via RNG or hashing). From our perspective, all these numbers have the same features, i.e., they are unforgeable, unrepeatable evidences that only the authenticator can generate. As a matter of fact, a user cannot perceive any observable difference between them. Therefore, we distinguish between data items only depending on their role in the protocol. In particular, here we consider three possible data items that one can infer from the behavior of the authenticator.

ε We use ε to denote that the authenticator receives/sends no data or when the transmitted data play no role in the authentication protocol.

`opid` An *operation identifier* is a piece of information that univocally defines the ongoing operation. It may be of different formats (e.g., a QR-code, an alphanumeric string) and it can be interpreted only by authenticators.

`otp` A *one-time password* is a code generated with some (assumed cryptographically perfect) algorithm that only the authorized parties can compute and verify. The internal structure of the `otp` depends on the inputs received by the authenticator that generates it. For instance, it can be the next number returned by a secure random number generator, a hash or the signature of an `opid`.

Example 3. *Both Nordea1 and Nordea2 terminate with the bank server receiving a response code. In the first case it amounts to the challenge answer. In the second case it is the confirmation code generated after the notification. In both cases we assume that the authenticators receive an identification code, i.e., `opid`, and return an encrypted confirmation code, i.e., `otp`.* □

3.2.4 Further Notation and Restrictions

The specification given in (3.1) provides a general definition of an authenticator. Moreover, we apply few restrictions to the structure of *well-formed* authenticators.

We already mentioned that a distinguishing feature of (software and hardware) authenticators is whether they inform the user about the ongoing operation. In general, these authenticators prompt the user with a message with the operation to be authorized, e.g., "transfer 100$ to account 1234". Then the user has to confirm the operation. This fact denotes the generation of an output (an `otp` in our setting) that is uniquely associated to a specific operation. Such an association is also called *dynamic linking* [Eur15, Eur17]. The definition of dynamic linking given there also states that the user must be aware of the ongoing operation and she has to explicitly agree on it. We use the label **?** to denote an authenticator type that informs the user and asks for her authorization as discussed above, e.g., 🖩[O]. When using such label, we assume that the user is able to properly check the received information and that she stops the protocol execution when such information does not match with the operation she wants to authorize. Reasonably, in order to display the operation details, the authenticator must receive an input, i.e., $\gamma \neq \varepsilon$. Well-formed authenticators must respect this requirement.

The Secret authenticator type consists of pure data objects, e.g., passwords or other information that can be stored on some support. As such, they do not have a proper input/output interaction (we assume that the user copies the information on some endpoint such as a web browser). For this reason 🔑 and ▦ are not annotated with channels and data items.

Furthermore, we introduce two abbreviations that will simplify the discussion about the out-of-band authenticators (see Section 3.4 below). In particular we define 🖩 $\triangleq$ 🖩[O] and 📱 $\triangleq$ 🔊[O]. Moreover, to apply these abbreviations two conditions must be satisfied:

- (🖩) $\gamma = m \vee \gamma' = m$; i.e., input or output are transmitted through mobile channels and,
- (📱) $\gamma = n \vee \gamma' = n$, i.e., input or output are transmitted through network channels.

Example 4. *We combine the observations of Examples 1, 2 and 3 to provide the specification for Nordea1 and Nordea2. The first protocol (Nordea1) corresponds to the following specification.*

$$🔑; \texttt{opid} \gg_h 🖩[\mathrm{O,K}] \gg_h \texttt{otp} \tag{3.2}$$

*For Nordea2, we replace 🔊[O,I] with the abbreviation introduced above, i.e., 📱[I]. Moreover, we notice that the authenticator informs the user about the ongoing operation (**?**). Thus, the resulting specification for the second protocol is as follows.*

$$🔑; \texttt{opid} \gg_n 📱^{?}[\mathrm{I}] \gg_n \texttt{otp} \tag{3.3}$$

□

Finally, we define two utility functions, i.e., *RunsOn* and *Endpoint*.

$RunsOn : A \rightarrow \{Desktop, Mobile, Hardware, \perp\}$ binds each authenticator to the platform where it runs. Intuitively, $RunsOn(a) = d$ means that the authenticator a is executed on a platform of type d. For instance $RunsOn(\texttt{opid} \gg_n \text{📱?}[\text{I}] \gg_n \texttt{otp}) = Mobile$. We use $\perp$ when a certain authenticator is not executed, e.g., $RunsOn(\text{🔑}) = \perp$.

Instead, $Endpoint : P \rightarrow \{Browser, App\}$ returns the user endpoint of a given protocol. For instance, $Endpoint(\text{🔑}; \texttt{opid} \gg_n \text{📱?}[\text{I}] \gg_n \texttt{otp}) = Browser$.

3.3 Semantics of SLaMP

In this section we present the semantics of SLaMP (presented in Section 3.2). We define the semantics in terms of its translation into a well-known state-of-the-art specification, namely SPS [AMV15]. Notice that this serves also as a basis for the implementation of a translator from our language for MFA protocol into the SPS specification.

3.3.1 Roles

In our specification, we assume that the participants to the MFA protocol executions, i.e., the Roles of the protocol, belong to a finite set. This set is composed as follows.

Endpoint (E) The Endpoint entity represents the browser or the application from which the protocol is started. One may consider it as a mere medium for transmitting data to the Server. In our modeling, instead, we use a specific role for this entity. This allows for better specifying the possible interactions and communications with the User, the Authenticators and the Server.

Depending from where the protocol is started, the endpoint could be of two types. In the case that a protocol is started from a desktop computer, we assume the Endpoint to be a browser installed on such computer. On the contrary, if a protocol is started from a mobile device, we assume that the endpoint is a native mobile application running on the smartphone of the user.

Authenticators ($\mathbf{A}_1$,..., $\mathbf{A}_n$) In our modeling, each authenticator employed in the protocol execution has a role on its own. This allows us to model all the possible communications of various types of authenticators (leveraging IPC, network, etc.) and specific interactions with the user.

It is worth noting that memorized and look-up secrets have not a role by themselves in the protocol. Being either memorized or calculated by the User, these two kinds of secret are modeled as a secret knowledge that is shared between the User and the Server.

User (U) In our modeling, the user has a specific role in the protocol execution. She is required to perform actions and interact with the endpoint and with the authenticators. In our modeling, we assume that she is the only one that can start the protocol execution. Moreover, we assume that she performs every action in a proper way, i.e., not making mistakes and not leaking any secret, authentication factor or authenticator output.

Server (S) The Server belongs to the online service on which the user wants to authenticate. We assume that any "client" (e.g., the Endpoint and every Authenticator) can authenticate the Server leveraging public certificates.

Telephony Server (T) For communicating through the mobile telephony network, the Server relies on an external Telephony Server or module. This server only communicates to the clients through the mobile telephony network and communicates with the Server for receiving instructions and transfer the data that has been sent by the authenticators.

3.3.2 Channels

Each role transmits and receives data through various communication channels. These channels can be of various kind, ranging from network-based channels (e.g., HTTPS connection) to "fictive" channels (e.g., the interactions between the authenticator and the user). In our modeling language, we consider two channels for every connection between two roles, depending on the "direction" of the data transmission. We generically indicate with *A to B* the communication channel through which the role *A* sends data to the role *B*, where *to* indicates the direction of the data transmission. Below we briefly introduce the channels appearing in our specifications as well as their technical features.

E to S, S to E These two channels model the communication between the Endpoint and the Server. Such communication can occur over a unilateral TLS channel, relying on a public certificate. Therefore, we assume **E to S** to be confidential only, while **S to E** is assumed to guarantee confidentiality and authenticity (E can authenticate S through its certificate).

S to A, A to S These two channels model the communication between the Server and an Authenticator. These channels are activated during the binding procedure, in which the shared secrets (including those for establishing a secure connection) are distributed in a controlled environment. For instance, A may share a symmetric key with S that has been generated during the process. For this reason, we assume that these channels are authentic and confidential.

U to A, U to A These two fictive channels model the interactions between the User and an Authenticator. We assume that the user keeps the authenticator carefully, being the only one who can use it. Additionally, we assume that, by default, every authenticator is employed in a "protected" environment, i.e., with no malicious agents eavesdropping data.

U to E, E to U These two fictive channels model the interactions between the User and the Endpoint. These communications take place through the screen and the keyboard of the device on which the protocol is executed. By default, we assume than any input of the user is confidential w.r.t. the Endpoint. On the other hand, we assume that the communication from E to U is both confidential and authentic (i.e., the Endpoint has a strict interaction with only one user).

E to A, A to E These two channels model a communication between the Endpoint and the Authenticator, e.g., leveraging an USB interface. In our modeling, we assume that such communications are performed leveraging specific USB drivers or inter-process-communication primitives that have been implemented in a proper way. In other words, we assume (by default) that these channels are protected from illegal accesses, being both confidential and authentic.

A to T, T to A This couple of channels represents the transmission of data between an Authenticator to the Telephony Server via the telephony network. In the past, there have been concerns on the security properties of the telephony network [MKTM16, MBSS13]. Nevertheless, here we assume it to be both confidential and authentic. The main reason is that attackers that can effectively compromise this two properties on the telephony network, although possible, are out of scope of this work.

T to S, S to T These two channels model the communications between the server and the Telephony Server. We assume that these communications are performed over a secure connection, guaranteeing confidentiality and mutual authentication (e.g., a VPN).

It is worth noting that a Data Channel specified in SLaMP (see Section 3.2.2) may refer to one or more of the presented communication channels. For instance, if an authenticator receives an input through the *optical* channel, **S to E** and **E to A** are involved, since the Server must send the input to the Endpoint in order for the authenticator to read it. On the contrary, if an authenticator sends the `otp` through the *network*, this maps to the only **A to S** channel.

3.3.3 SPS specification

We have defined the semantics of our language in terms of Security Protocol Specification language [AMV15] (SPS, from now on), a variant of existing Alice-and-Bob languages. Indeed, SPS specifies various additional features with respect to these languages (e.g., communication channels), covering all the aspects that we consider in our framework.

The syntax of SPS is reported in EBNF form in Listing 3.1. An SPS specification relies on six sections.

```
SPS ::= Types :    (TYPE IDENT_s; )*
        Mappings :   (FUNC : TYPE_s → TYPE; )*
        Formats :    (FUNC(TYPE_s); )*
        Knowledge :    (ROLE : MSG_s;)*[ where ROLE ≠ ROLE ( & ROLE ≠ ROLE )*]
        Actions :    (ROLE CHANNEL ROLE : MSG | ROLE : TYPE VAR)*
        Goals :    (ROLE authenticates ROLE ON MSG | MSG secret of ROLE_s )*
MSG ::= CONST | VAR | FUNC(MSG_s)
IDENT ::= CONST | VAR | FUNC
ROLE ::= CONST | VAR
TYPE ::= Agent | Number | PublicKey | PrivateKey | SymmetricKey | Bool | Msg
CHANNEL ::= [ • ] → [ • ]
```

Listing 3.1: EBNF of SPS specification.

In the **Types** section, all constants and variables are declared, using a set of pre-defined types. Among these types, the type **Agent** is used to identify Roles, i.e., the participants to the protocol; the **Number** type is used to indicate a generic fresh number/value; other types define various kinds of keys, boolean values and the generic **Msg** type, subsuming all types. The IDENTIFIER of each specified type can be a constant (CONST), a function (FUNC) or a variable (VAR); CONST and FUNC are alphanumeric strings starting with a lower-case letter, while VAR is an alphanumeric string starting with an upper-case letter.

In the **Mappings** section, it is possible to specify a special kind of function symbols that can be used to describe pre-existing setup of long-term keys. In this section, it is hence possible to define shared keys for agents and public key infrastructures.

In the **Formats** section, one can specify function symbols that basically represents a concatenation of information. Formats are used to abstractly represent how the concrete implementation structures the clear-text part of a message, allowing to generate (in the SPS translation to formal models) implementations with real-world formats such as TLS.

In the **Knowledge** section, it is possible to specify the initial knowledge of each of the protocol roles. For instance, it is possible to declare the knowledge of shared secrets or other principals.

The **Actions** section contains both the creation of new fresh values and the messages that are exchanged between the roles. In the first case, the action contains the role creating the variable along with the type and the name of the variable itself. In the latter, the single line includes a the two roles involved in the communication, the employed **Channel** and the transmitted data (**Msg**). In particular, the type CHANNEL allows for specifying authentic, confidential, and secure channels as $\bullet \rightarrow$, $\rightarrow \bullet$ and $\bullet \rightarrow \bullet$, respectively. Instead, the transmitted data can be constituted of constants, variables and functions. Among these functions, a set of fixed cryptographic function symbols (included in SPS by default) can be indicated. These functions are asymmetric and symmetric encryption (`crypt` and `scrypt`), digital signatures (`sign`), hash and keyed-hash functions (`hash` and `mac`), and modular exponentiation (`exp`) and multiplication (`mult`). It is worth noting

```
1  Types :
2          Agent A, B;
3          Number g;
4
5  Mappings :
6          shk : Agent, Agent -> SymmetricKey ;
7
8  Formats :
9          f1(Agent, Agent, Msg);
10         f2(Number);
11
12 Knowledge :
13         A: A, B, shk(A,B), g;
14         B: B, A, shk(A,B), g;
15
16 Actions :
17         A: Number X
18         A -> B : scrypt(shk(A,B), f1(A,B,exp(g,X)))
19         B: Number Y
20         B -> A : scrypt(shk(A,B), f1(B,A,exp(g,Y)))
21         A: Number Payload
22         A -> B : scrypt(exp(exp(g,Y),X), f2(Payload))
23
24 Goals :
25         Payload secret of A,B
```

Listing 3.2: Example of protocol specification in SPS.

that, as a consequence of the specified actions, some implicit operations may take place. For instance, when receiving an encrypted message, an agent would implicitly decrypt it and keep it in its knowledge, if it is in possession of the proper key. Moreover, after receiving a known variable, an agent would implicitly check its value and stop further actions if the received value is not the one expected.

Finally, the **`Goals`** section includes the goals the protocol aims to achieve, i.e., secrecy and authentication goals.

An example of SPS specification of a protocol is reported in Listing 3.2. In the specified protocol, two agents A and B use a symmetric key `shk(A, B)` to establish a fresh Diffie-Hellman key and securely exchange a Payload message. The `Types` section (lines 1-3) includes the specification of the two agents (`A`,`B`) and the constant value `g`. The `Mappings` section contains the structure of the shared key (lines 5-6). The `Formats` section includes the specification of two message formats, for transmitting the half-key from an agent to the other (line 9) and the payload (line 10). The `Knowledge` section contains the specification of the initial knowledge of both the agents, sharing the (secret) symmetric key and a g value (lines 13 ad 14, respectively). The `Actions` section

contains the messages exchanged by the agents and their internal operations. A creates the secret exponent X for the Diffie-Hellman exchange (line 17). Then, it computes the half-key `exp(g, X)`, inserts it into format `f1`, encrypts the message with the shared key `shk(A, B)` and sends it to B leveraging a standard insecure channel (line 18). Since the encryption key is shared, B is able to decrypt the message and obtain the content in `f1`. B does the same actions as A: it creates the secret exponent Y (line 19), it computes the half-key `exp(g, X)`, it inserts it into format `f1`, encrypts the message with the shared key `shk(A, B)` and sends it to A leveraging the channel in the opposite direction (line 20). At this point, A generates a generic Payload (line 21). Finally, A inserts the Payload into format `f2`, encrypts the message with the "composed" key, `exp(exp(g,Y),X)`, and sends it to B (line 21). B is able to decrypt the message, combining the known variables and obtaining the `Payload`.

Given an SPS specification σ, we say that it is *valid*, in symbols $\vdash \sigma$ if none of its goals are violated. Otherwise, we say that σ is invalid ($\nvdash \sigma$).

3.3.4 Translation into SPS

In this section, we define the SPS semantics of SLaMP. The semantics is given in terms of the function Sem : SLaMP$\rightarrow$ *SPS*.

The definition of the *Sem* function requires us to introduce a number of utility functions. Each utility function is responsible for building a fragment of the final specification. All the fragments are then composed together by *Sem*. In the following sections we describe these functions, then, in Section 3.3.4.7, we provide the definition of *Sem*.

3.3.4.1 Actions Function

We start by defining the function *Actions*, that is responsible for building the list of SPS actions out of an SLaMP specification. In symbols, $Actions(P) = Actions(A_1;\ldots;A_n) = PRE \cdot Actions_1(A_1) \cdot \ldots \cdot Actions_n(A_n)$, where (i) *PRE* is a fixed list of SPS actions (given in Listings 3.3), and (ii) the function $Actions_k(A_k)$ is defined in Section 3.3.4.1.1. The symbol $\cdot$ denotes the concatenation between two lists.

```
U: Msg Operation
U →• E: Operation
E →• S: Operation
```

Listing 3.3: Actions of *PRE*.

Intuitively, *PRE* models the list of actions that must be executed at the start of every protocol. Indeed, in our modeling, an MFA protocol starts when the User decides to perform an `Operation` on the Server. Such operation is firstly created as a fresh value, as specified in the first line of Listing 3.3. Then, the `Operation` is inserted by the User on her Endpoint (from which we assume she starts the protocol execution) - see line 2. Such operation request is then forwarded from the Endpoint to the Server (line 3). At this point, the operations and interactions with the authenticators

will start.

3.3.4.1.1 Actions($A_1;\ldots;A_n$) The translation of the authenticators to actions is modeled with the function *Actions*, defined as

$$Actions(A_1;\ldots;A_n) = PRE \cdot Actions_1(A_1) \cdot \ldots \cdot Actions_n(A_n)$$

without risk of misunderstanding, we use $Actions_k$ to denote the application of function *Actions* on the k-th authenticator.

Again, $Actions_k$ amounts to the concatenation of the results of three functions, i.e., $Actions_k^{in}$, $Actions_k^{mid}$ and $Actions_k^{out}$, that we define below. In symbols, $Actions_k(\delta \gg_\gamma \tau^{(?)}[F] \gg_{\gamma'} \delta') = Actions_k^{in}(\delta \gg_\gamma \tau^{(?)}[F] \gg_{\gamma'} \delta') \cdot Actions_k^{mid}(\delta \gg_\gamma \tau^{(?)}[F] \gg_{\gamma'} \delta') \cdot Actions_k^{out}(\delta \gg_\gamma \tau^{(?)}[F] \gg_{\gamma'} \delta')$.

Definition of $Actions_k^{in}$. This function returns the list of actions corresponding to the input of a given authenticator. The extensional definition of $Actions_k^{in}$ is given in Table 3.2. in the following, further details on possible values of $Actions_k^{in}$ are given.

$Actions_k^{in}(\texttt{opid} \gg_h \tau^{(?)}\textbf{[F]} \gg_{\gamma'} \delta')$ **where** $O \in F$**.** Indicates a list of actions allowing the Server (S) to transmit `opid` to the Authenticator (A_k) relying on the manual copy of the user. Obviously, the manual input of the user is consequent to a set of communications and operations that take place before that particular action. It is worth noting that, in this work, we assume that an `opid` is based on two elements. Firstly, we assume that an `opid` contains a reference to the `Operation` that the User has communicated to the Server. Secondly, we assume that it is associated to fresh value, for preventing potential replay attacks. Therefore, the first operation of the list indicates that the Server generates a fresh, unique identifier (`UniqueID`) of the Operation.

The `opid`, expressed as a format containing the combination of `Operation` and `UniqueID`, is then sent from the Server to the Endpoint. As mentioned in Section 3.2.3, we assume that the `opid` is created for (and can be read by) the specific authenticator. This fact is modeled by using a symmetric encryption function (`scrypt` - one of the base functions of SPS) for encrypting the `opid` with a specific decoding key $dK(S,A_k)$, which is shared between the Authenticator and the Server. At this point, the User takes `opid` from the Endpoint (this action is modeled as a transmission of the encrypted `opid` from E to U) and she inserts it into the Authenticator.

Table 3.2: Extensional definition of $Actions_k^{in}$.

Input	Output
$\texttt{opid} \gg_h \tau^{(?)}[\text{F}] \gg_{\gamma'} \delta'$ where $O \in F$	S : Number UniqueID S • → • E: scrypt(dK(S,A_k), opid(Operation,UniqueID)) E • → • U: scrypt(dK(S,A_k), opid(Operation,UniqueID)) U • → • A_k: scrypt(dK(S,A_k), opid(Operation,UniqueID))
$\texttt{opid} \gg_h \tau^{(?)}[F] \gg_{\gamma'} \delta'$ where $O \notin F$	S : Number UniqueID S • → • E: scrypt(dK(S,A_k), opid(Operation,UniqueID)) E • → • U: scrypt(dK(S,A_k), opid(Operation,UniqueID)) U → • A_k: scrypt(dK(S,A_k), opid(Operation,UniqueID))
$\texttt{opid} \gg_o \tau^{(?)}[\text{F}] \gg_{\gamma'} \delta'$ where $O \in F$	S : Number UniqueID S • → • E: scrypt(dK(S,A_k), opid(Operation,UniqueID)) E • → • A_k: scrypt(dK(S,A_k), opid(Operation,UniqueID))
$\texttt{opid} \gg_o \tau^{(?)}[\text{F}] \gg_{\gamma'} \delta'$ where $O \notin F$	S : Number UniqueID S • → • E: scrypt(dK(S,A_k), opid(Operation,UniqueID)) E → • A_k: scrypt(dK(S,A_k), opid(Operation,UniqueID))
$\texttt{opid} \gg_n \tau^{(?)}[\text{F}] \gg_{\gamma'} \delta'$	S : Number UniqueID S • → • A_k: scrypt(dK(S,A_k), opid(Operation,UniqueID))
$\texttt{opid} \gg_i \tau^{(?)}[\text{F}] \gg_{\gamma'} \delta'$ where $O \in F$	S : Number UniqueID S • → • E: scrypt(dK(S,A_k), opid(Operation,UniqueID)) E • → • A_k: scrypt(dK(S,A_k), opid(Operation,UniqueID))
$\texttt{opid} \gg_i \tau^{(?)}[\text{F}] \gg_{\gamma'} \delta'$ where $O \notin F$	S : Number UniqueID S • → • E: scrypt(dK(S,A_k), opid(Operation,UniqueID)) E → • A_k: scrypt(dK(S,A_k), opid(Operation,UniqueID))
$\texttt{opid} \gg_m \tau^{(?)}[\text{F}] \gg_{\gamma'} \delta'$	S : Number UniqueID S • → • T: scrypt(dK(S,A_k), opid(Operation,UniqueID)) T • → • A_k: scrypt(dK(S,A_k), opid(Operation,UniqueID))
$\texttt{otp} \gg_h \tau[\text{F}] \gg_{\gamma'} \delta'$ where $O \in F$	S • → • E: hash(seed(S,A_k)) E • → • U: hash(seed(S,A_k)) U • → • A_k: hash(seed(S,A_k))
$\texttt{otp} \gg_h \tau[\text{F}] \gg_{\gamma'} \delta'$ where $O \notin F$	S • → • E: hash(seed(S,A_k)) E • → • U: hash(seed(S,A_k)) U → • A_k: hash(seed(S,A_k))
$\texttt{otp} \gg_m \tau[\text{F}] \gg_{\gamma'} \delta'$	S • → • T: hash(seed(S,A_k)) T • → • A_k: hash(seed(S,A_k))
$\texttt{otp} \gg_m \tau^{?}[\text{F}] \gg_{\gamma'} \delta'$	S: Number UniqueID S • → • T: crypt(pK(A_k),scrypt(dK(S,A_k),opid(Operation,UniqueID))) T • → • A_k: crypt(pK(A_k),scrypt(dK(S,A_k),opid(Operation,UniqueID)))

$Actions_k^{in}(\texttt{opid} \gg_h \tau^{(?)}\textbf{[F]} \gg_{\gamma'} \delta')$ **where** $O \notin F$**.** This case refers to the possibility that an authenticator, receiving an input via a human, does not attest an Ownership factor (e.g., gets stolen). The list of actions remains the same w.r.t. the case with the ownership factor, except for the last one, in which the channel between the User and the Authenticator loses the authenticity property.

$Actions_k^{in}(\texttt{opid} \gg_o \tau^{(?)}\textbf{[F]} \gg_{\gamma'} \delta')$ **where** $O \in F$**.** This is the case in which the Authenticator requires to scan a barcode/QR code in order read the input provided by the Server. It is worth noting that the scanning operation is usually performed on a display, hence - in this case - `opid` is a sort of a barcode/QR code to be shown on the Endpoint. The list of operations starts with the Server generating a unique identifier (`UniqueID`) related to the specific Operation. Consequently, the Server implicitly generates the `opid` containing `Operation` and `UniqueID`. As for the previous case, the `opid` must be read only by the specific Authenticator with the proper decoding key. Hence, the Server sends the `opid`, encrypted with $\texttt{dK(S,A}_\texttt{k}\texttt{)}$ (a decoding key shared with the authenticator) to the Endpoint. The Authenticator has to scan the `opid` from the display of the endpoint. This action is modeled as a transmission of `opid` from the Endpoint to the Authenticator, that implicitly decrypts it, leveraging the symmetric key $\texttt{dK(S,A}_\texttt{k}\texttt{)}$.

$Actions_k^{in}(\texttt{opid} \gg_o \tau^{(?)}\textbf{[F]} \gg_{\gamma'} \delta')$ **where** $O \notin F$**.** In this case, we refer to an authenticator that does not attest an Ownership factor and receives an input by scanning a code from the Enpoint. The list of actions remains the same w.r.t. the case in which $O \in F$, with the exception of the last one, in which the channel between the Endpoint and the Authenticator loses the authenticity property.

$Actions_k^{in}(\texttt{opid} \gg_n \tau^{(?)}\textbf{[F]} \gg_{\gamma'} \delta')$**.** This case models the transmission of the `opid` from the Server to the Authenticator leveraging a the network. For this input of $Actions_k^{in}$, the resulting list of actions is constituted by an internal operation and a data transmission. At first, the Server creates the `UniqueID`. Then, the Server sends the `opid` (encrypted with the decoding key $\texttt{dK(S,A}_\texttt{k}\texttt{)}$) to the Authenticator.

$Actions_k^{in}(\texttt{opid} \gg_i \tau^{(?)}\textbf{[F]} \gg_{\gamma'} \delta')$ **where** $O \in F$**.** This is the case in which the Authenticator is directly connected to the endpoint, either leveraging an USB interface (if it is an hardware device) or a direct communication between the software application and the endpoint. The corresponding list of actions starts with an internal operation, through which the Server generates a `UniqueID` related to the chosen operation. The `opid` is then sent from the Server to the Endpoint, encrypted with the decoding key $\texttt{dK(S,A}_\texttt{k}\texttt{)}$. At this point, the Endpoint transmits the obtained `opid` to the connected Authenticator.

$Actions_k^{in}(\texttt{opid} \gg_i \tau^{(?)}\textbf{[F]} \gg_{\gamma'} \delta')$ **where** $O \notin F$**.** In this case, we refer to an authenticator which is connected to the endpoint, receives an input via USB or IPC and, does not attest an Ownership factor. The list of actions remains the same w.r.t. the "standard" case, with the

exception of the last one, in which the channel between the Endpoint and the Authenticator loses the authenticity property.

$Actions_k^{in}(\texttt{opid} \gg_m \tau^{(?)}\textbf{[F]} \gg_{\gamma'} \delta')$**.** In this case, the Server contacts the Authenticator through the telephony network, leveraging the Telephony Server. It is assumed that the Service Provider has enough information to establish a communication with the correct authenticator, i.e., the one associated to the user's identity (in the case of an Out-of-Band Hardware authenticator, we assume that the Server knows the phone number associated to the user's account).

The first operation is internal: the Server generates an `UniqueID` associated to the operation chosen by the user. Then, the Server contacts the Telephony Server, that could reach authenticators through the mobile telephony network. The Server transmits the `opid` to the Telephony Server, encrypted with the decoding key which has been shared with the Authenticator ($\texttt{dK(S,A}_\texttt{k}\texttt{)}$). The Telephony Server finally sends the `opid` to the correct Authenticator, through the telephony network.

$Actions_k^{in}(\texttt{otp} \gg_h \tau\textbf{[F]} \gg_{\gamma'} \delta')$ **where** $O \in F$**.** In this case, the Server transmits an `otp` to the Endpoint, requiring the user to copy it and put it on the Authenticator. The Server firstly generates an `otp`, not depending from an `opid`. The structure of this `otp` may be of the most various types. However, an `otp` - by its nature - has to be a fresh value of limited validity over time. Therefore, we model it as a value which has been generated depending on a seed that has been previously shared between the Server and the Authenticator. Hence, the first action of the list, represent the Server sending the `otp` to the Endpoint. The `otp` is generated using the hash function (as suggested by [NIS17] for the generic single-factor authenticator), taking $\texttt{seed(S,A}_\texttt{k}\texttt{)}$ as an input (as mentioned in Section 3.3.4.5, we assume that the seed has been correctly distributed from the Server to the specific Authenticator k). Then, the User reads the `otp` from the Endpoint. This is modeled with a transmission from E to U. Finally, the User inserts the `otp` on the Authenticator (transmission from U to $\texttt{A}_\texttt{k}$).

$Actions_k^{in}(\texttt{otp} \gg_h \tau\textbf{[F]} \gg_{\gamma'} \delta')$ **where** $O \notin F$**.** This case refers to an authenticator, receiving an `otp` by manual insertion, that does not attest an Ownership factor. The list of actions remains the same w.r.t. the "standard" case, with the exception of the last one, in which the channel between the Endpoint and the Authenticator loses the authenticity property.

$Actions_k^{in}(\texttt{otp} \gg_m \tau\textbf{[F]} \gg_{\gamma'} \delta')$**.** In this case, the Server sends an `otp` to the Authenticator leveraging the Telephony Server. As for the previous case, the Server generates an `otp` through a `hash` function, taking the shared `seed` as an input. Then, it sends it to the Telephony Server (first action of the list). Finally, the Telephony Server send the `otp` to the Authenticator.

$Actions_k^{in}(\texttt{otp} \gg_m \tau^{\textbf{?}}\textbf{[F]} \gg_{\gamma'} \delta')$**.** In this case, the Server for transmits an `otp` to the Authenticator leveraging the Telephony Server. However, since the Authenticator displays the ongoing information, this means that the `opid` has also to be transmitted. In this particular case, the

Table 3.3: Extensional definition of $Actions_k^{mid}$.

Input	**Output**
$\delta \gg_\gamma \tau[\{K\} \cup F] \gg_{\gamma'} \delta'$	U ● → ● A_k: fK(U,A_k)
$\delta \gg_\gamma \tau[\{I\} \cup F] \gg_{\gamma'} \delta'$	U ● → ● A_k: fI(U,A_k)
$\delta \gg_\gamma \tau[\{K,I\} \cup F] \gg_{\gamma'} \delta'$	U ● → ● A_k: fK(U,A_k) U ● → ● A_k: fI(U,A_k)
$\delta \gg_\gamma \tau^?[\mathrm{O}] \gg_{\gamma'} \delta'$	A_k ● → ● U: Operation
$\delta \gg_\gamma \tau^?[\{K\} \cup F] \gg_{\gamma'} \delta'$	U ● → ● A_k: fK(U,A_k) A_k ● → ● U: Operation
$\delta \gg_\gamma \tau^?[\{I\} \cup F] \gg_{\gamma'} \delta'$	U ● → ● A_k: fI(U,A_k) A_k ● → ● U: Operation
$\delta \gg_\gamma \tau^?[\{K,I\} \cup F] \gg_{\gamma'} \delta'$	U ● → ● A_k: fK(U,A_k) U ● → ● A_k: fI(U,A_k) A_k ● → ● U: Operation
🔑	U → ● E: mem(S,U,k) E → ● S: mem(S,U,k)
▦	S : Number Coords S ● → ● E: Coords E ● → ● U: Coords U ● → ● A_k: Coords A_k ● → ● U: exp(lus(S,A_k),Coords) U → ● E: exp(lus(S,A_k),Coords) E → ● S: exp(lus(S,A_k),Coords)
otherwise	$\emptyset$

Server generates an otp that contains an opid (that can only be decoded with the proper dK key). Then, it creates an otp, signing it with the public key of the Authenticator k, and sends such a message to the Telephony Server. The Telephony Server finally transmits the otp to the Authenticator, leveraging the mobile telephony network. The Authenticator can implicitly decode the message and get the information regarding the Operation that has to be displayed to the user.

Definition of $Actions_k^{mid}(\tau[\mathbf{F}])$. This function returns the list of actions deriving from the type and the set of attested authentication factors of a given authenticator. The extensional definition of this function is presented in Table 3.3. In the following paragraph, further details of possible values of the functions are presented.

$Actions_k^{mid}(\tau[\{K\} \cup \mathbf{F}])$**.** In the case that an authenticator testifies an additional Knowledge factor, an additional action must be considered. This action models the interaction of the User

(U) with the Authenticator (A_k), providing the knowledge factor (`fK(U,`A_k`)`) in order to unlock the functionality of the Authenticator. If the received knowledge factor is correct, the rest of the actions (appended after this one by the function *Actions*) will be executed.

$Actions_k^{mid}(\tau[\{I\}\cup\mathbf{F}])$**.** As for the previous case, when an Authenticator testifies an additional Inherence factor, an additional action has to be considered: the interaction of the User (U) with the Authenticator (A_k), providing the inherence factor (`fI(U,`A_k`)`) in order to unlock the functionality of the Authenticator.

$Actions_k^{mid}(\tau[\{K,I\}\cup\mathbf{F}])$**.** When an Authenticator testifies additional Knowledge and Inherence factors, two actions modeling the interaction of the User (U) with the Authenticator (A_k), providing both the knowledge and the inherence factors (`fK(U,`A_k`)`, `fI(U,`A_k`)`) must be included.

$Actions_k^{mid}(\tau^{?}[\mathbf{O}])$**.** A single-factor Authenticator, displaying information regarding the ongoing operation, implicitly requires an additional action. The action specifies a communication from the Authenticator (A_k) to the User (U), which models the fact the User is shown the critical information regarding the operation that she is about to authorize. At this point, since the variable `Operation` has been initially created by the User, an implicit check such variable is performed. If the value of `Operation` is the one expected, i.e., it belongs to the knowledge of the user, the rest of the actions are executed. Instead, if the operation displayed to the user does not match with what she is expecting, the protocol should stop: by continuing the protocol execution, indeed, another (potentially malicious) operation may be authorized.

$Actions_k^{mid}(\tau^{?}[\{K\}\cup\mathbf{F}])$**.** In the case that an Authenticator, attesting an additional Knowledge factor, displays information regarding the ongoing operation, two actions are required. These two actions model *a)* the transmission from the User to the Authenticator of the knowledge factor and *b)* the fact that the Authenticator displays to the User the information about the operation. In this case, two implicit checks on `fK` and `Operation` are executed by the Authenticator and the User, respectively.

$Actions_k^{mid}(\tau^{?}[\{I\}\cup\mathbf{F}])$**.** As for the previous case, if an Authenticator, attesting an additional Inherence factor, displays information regarding the ongoing operation, two actions are required. Trivially, these two actions model *a)* the transmission from the User to the Authenticator of the inherence factor and *b)* the fact that the Authenticator displays to the User the information about the operation. The values of `fI` and `Operation` must be checked in order to proceed with the correct execution of the protocol.

$Actions_k^{mid}(\tau^{?}[\{K,I\}\cup\mathbf{F}])$**.** In the case that an Authenticator, attesting additional Knowledge and Inherence factors, displays information regarding the ongoing operation, two actions are required. Trivially, these three actions model *a)* the transmission from the User to the

Authenticator of both the knowledge and inherence factors and *b)* the fact that the Authenticator displays to the User the information about the operation. Also in this case, the values of the variables have to be properly checked.

$Actions_k^{mid}$(🔑)**.** When considering 🔑, a particular list of actions must be included in the specification. Indeed, since 🔑 is modeled as a shared knowledge, the list would start with the User inserting the $k-th$ memorized secret on the Endpoint. The Endpoint will forward the memorized secret to the Server. If the received memorized secret is correct, the actions following this list will be executed.

$Actions_k^{mid}$(▦)**.** In the case of ▦, a particular list of actions must be included. The Authenticator is modeled as a device having a set of pre-shared values (`lus`). The list of actions starts with the Server generating a set of coordinates for retrieving the correct secret (modeled as a fresh variable `Coordinates`). The `Coordinates` are sent to the Endpoint. At this point, the User reads the coordinates from the Endpoint (transmission from E to U) and "puts" them on the Authenticator (transmission from U to A_k). Leveraging the `lus` mapping and the coordinates, a value is returned, obtained as a math calculus. Here we model it with the function `exp`. The User inserts the obtained secret to the Endpoint. The Endpoint sends it to the Server, leveraging an HTTPS channel. If the received secret is correct, the actions following this list will be executed.

Definition of $Actions_k^{out}$. This function generates the list of actions representing the operations, interactions and transmissions that allow the Authenticator (A) to transmit the generated output to the Server (S). The extensional definition of this function is presented in Table 3.4. In the following, we give further detail for every possible input of $Actions_k^{out}$.

$Actions_k^{out}(\tau\textbf{[F]} \gg_h \texttt{otp})$ **where** $O \in F$**.** In this case, the Authenticator communicates the `otp` to the Server leveraging some human actions. The list of actions starts with the Authenticator (A_k) generating an `otp` without receiving any input. As previously mentioned, we consider this kind of `otp` to be generated by a hash function, which takes `seed(S,`A_k`)` (i.e., a seed that has been previously shared between the Server and the Authenticator k) as input. The generated `otp` is read by the User (in the first action). Then, the `otp` is copied by the user on the Endpoint. Finally, the Endpoint sends it to the Server for verification. If the value of the received `otp` is as expected by the Server, the actions following this list will be executed.

$Actions_k^{out}(\tau\textbf{[F]} \gg_h \texttt{otp})$ **where** $O \notin F$**.** This case refers an authenticator generating (without any `opid` as input) an `otp` to be manually copied that does not attest an Ownership factor. Except for the first action, in which the channel between the Authenticator and the User is not confidential, the set of actions is identical w.r.t. the case in which $O \in F$.

Table 3.4: Extensional definition of $Actions_k^{out}$.

Input	Output
τ[F] $\gg_h$ otp where $O \in F$	$A_k \bullet\!\!\rightarrow\!\!\bullet$ U: hash(seed(S,A_k)) U $\rightarrow\!\!\bullet$ E: hash(seed(S,A_k)) E $\rightarrow\!\!\bullet$ S: hash(seed(S,A_k))
τ[F] $\gg_h$ otp where $O \notin F$	$A_k \bullet\!\!\rightarrow$ U: hash(seed(S,A_k)) U $\rightarrow\!\!\bullet$ E: hash(seed(S,A_k)) E $\rightarrow\!\!\bullet$ S: hash(seed(S,A_k))
τ[F] $\gg_n$ otp	$A_k \bullet\!\!\rightarrow\!\!\bullet$ S: hash(seed(S,A_k))
τ[F] $\gg_i$ otp where $O \in F$	$A_k \bullet\!\!\rightarrow\!\!\bullet$ E: hash(seed(S,A_k)) E $\rightarrow\!\!\bullet$ S: hash(seed(S,A_k))
τ[F] $\gg_i$ otp where $O \notin F$	$A_k \bullet\!\!\rightarrow$ E: hash(seed(S,A_k)) E $\rightarrow\!\!\bullet$ S: hash(seed(S,A_k))
τ[F] $\gg_m$ otp	$A_k \bullet\!\!\rightarrow\!\!\bullet$ T: hash(seed(S,A_k)) T $\bullet\!\!\rightarrow\!\!\bullet$ S: pair(A_k,hash(seed(S,A_k)))
opid$\gg_\gamma \tau^{(?)}$[F] $\gg_h$ otp where $O \in F$	$A_k \bullet\!\!\rightarrow\!\!\bullet$ U: crypt(inv(pK(A_k)),opid(Operation,UniqueID)) U $\rightarrow\!\!\bullet$ E: crypt(inv(pK(A_k)),opid(Operation,UniqueID)) E $\rightarrow\!\!\bullet$ S: crypt(inv(pK(A_k)),opid(Operation,UniqueID))
opid$\gg_\gamma \tau^{(?)}$[F] $\gg_h$ otp where $O \notin F$	$A_k \bullet\!\!\rightarrow$ U: crypt(inv(pK(A_k)),opid(Operation,UniqueID)) U $\rightarrow\!\!\bullet$ E: crypt(inv(pK(A_k)),opid(Operation,UniqueID)) E $\rightarrow\!\!\bullet$ S: crypt(inv(pK(A_k)),opid(Operation,UniqueID))
opid$\gg_\gamma \tau^{(?)}$[F] $\gg_n$ otp	$A_k \bullet\!\!\rightarrow\!\!\bullet$ S: crypt(inv(pK(A_k)),opid(Operation,UniqueID))
opid$\gg_\gamma \tau^{(?)}$[F] $\gg_i$ otp where $O \in F$	$A_k \bullet\!\!\rightarrow\!\!\bullet$ E: crypt(inv(pK(A_k)),opid(Operation,UniqueID)) E $\rightarrow\!\!\bullet$ S: crypt(inv(pK(A_k)),opid(Operation,UniqueID))
opid$\gg_\gamma \tau^{(?)}$[F] $\gg_i$ otp where $O \notin F$	$A_k \bullet\!\!\rightarrow$ E: crypt(inv(pK(A_k)),opid(Operation,UniqueID)) E $\rightarrow\!\!\bullet$ S: crypt(inv(pK(A_k)),opid(Operation,UniqueID))
opid$\gg_\gamma \tau^{(?)}$[F] $\gg_m$ otp	$A_k \bullet\!\!\rightarrow\!\!\bullet$ T: crypt(inv(pK(A_k)),opid(Operation,UniqueID)) T $\bullet\!\!\rightarrow\!\!\bullet$ S: pair(A_k,crypt(inv(pK(A_k)),opid(Operation,UniqueID)))
otp$\gg_\gamma \tau$[F] $\gg_m$ otp	$A_k \bullet\!\!\rightarrow\!\!\bullet$ T: hash(seed(S,A_k)) T $\bullet\!\!\rightarrow\!\!\bullet$ S: pair(A_k,hash(seed(S,A_k)))
otp$\gg_\gamma \tau$[F] $\gg_n$ otp	$A_k \bullet\!\!\rightarrow\!\!\bullet$ S: hash(seed(S,A_k))
otp$\gg_\gamma \tau^{?}$[F] $\gg_m$ otp	$A_k \bullet\!\!\rightarrow\!\!\bullet$ T: crypt(inv(pK(A_k)),opid(Operation,UniqueID)) T $\bullet\!\!\rightarrow\!\!\bullet$ S: pair(A_k,crypt(inv(pK(A_k)),opid(Operation,UniqueID)))

$Actions_k^{out}(\tau$**[F]** $\gg_n$ **otp**$)$**.** In this case, the Authenticator directly communicates with the Server. The Authenticator generates the otp with the hash function, leveraging seed(S,A_k), and sends it to the Server for verification. It is assumed that the communication takes place

after a previous establishment of a session between the Authenticator and the Server. If the value of the received `otp` is as expected by the Server, the actions following this list will be executed.

$Actions_k^{out}(\tau$**[F]** $\gg_i$ **otp**$)$ **where** $O \in F$**.** In this case, the Authenticator sends the `otp` to the Server, passing through the Endpoint by using IPC. The list starts with the Authenticator (A_k) generating the `otp` (with the `hash` function) and sending it to the Endpoint (E). At this point, the Endpoint transmits the `otp` to the Server. If the the verification of the received otp_T is positive, the actions following this list will be executed.

$Actions_k^{out}(\tau$**[F]** $\gg_i$ **otp**$)$ **where** $O \notin F$**.** This case refers to an authenticator that generates (without any `opid` as input) an `otp`, transmits it via IPC or USB interface while not attesting an Ownership factor. Except for the first action, in which the channel between the Authenticator and the Endpoint is not confidential, the set of actions is identical w.r.t. the case in which $O \in F$.

$Actions_k^{out}(\tau$**[F]** $\gg_m$ **otp**$)$**.** In this case, the Authenticator sends the `otp` through the telephony network for reaching the Server verification. To this aim, the Authenticator (A_k) generates the `otp` and sends it to the Telephony Server. The Telephony Server transmits the received `otp` to the Server, together with the identity of the sender. To do so, it leverages the format `pair` that, by definition, allows to transmit an Agent and a message. Therefore, the Telephony Server sends `pair(`A_k`, hash(seed(S,`A_k`))` to the Server. If the (implicit) verification of the received `otp` for the given Authenticator A_k returns a positive result, the actions following this list will be executed.

$Actions_k^{out}($**opid**$\gg_\gamma \tau^{(?)}$**[F]** $\gg_h$ **otp**$)$ **where** $O \in F$**.** In this case, an Authenticator, receiving an `opid` as input, generates an `otp` that has to be manually copied to the user. The list of actions starts with the Authenticator (A_k) that implicitly generates an `otp` that depends on the `opid`. As previously mentioned, the `opid` contains information regarding the `Operation` chosen by the User and a `UniqueID`, associated to the operation and given by the Server. As mentioned in Section 3.2, we assume that if the Authenticator receives an `opid` as an input, the generated `otp` must be somehow related to it. Hence, we model this kind of `otp` as the result of an asymmetric encryption, performed with the private key of the Autheticator (`inv(pK(`A_k`))`), of the `opid`. It is worth noting that the non-repeatability of the message is guaranteed by the data inserted in the `opid` itself. Therefore, the first action of the list indicates that the Authenticator displays the `otp` (modeled as as the asymmetric encryption of the `opid`) to the User. The `otp` is then copied by the user on the Endpoint (second actions). Finally, the Endpoint sends it to the Server for verification. The Server implicitly decrypts the obtained message with the public key (`pK(`A_k`)`) of the Authenticator. If the value of the received `otp` is as expected by the Server, the actions following this list will be executed.

$Actions_k^{out}(\mathbf{opid} \gg_\gamma \tau^{(?)}\mathbf{[F]} \gg_h \mathbf{otp})$ **where** $O \notin F$**.** This case refers to an authenticator, generating an `otp` (depending from an `opid`) to be copied by the user, that does not attest an Ownership factor. Except for the first action, in which the channel between the Authenticator and the User is not confidential, the set of actions is identical w.r.t. the case in which $O \in F$.

$Actions_k^{out}(\mathbf{opid} \gg_\gamma \tau^{(?)}\mathbf{[F]} \gg_n \mathbf{otp})$**.** In this case, an Authenticator, receiving an `opid` as input, generates an `otp` that is sent to the Server from the Authenticator itself.

The Authenticator (A_k) generates a the `otp` by encrypting the `opid` with its private key (written as `crypt(inv(pK(`A_k`)),opid(Operation,UniqueID))`) and sends it to the Server (S). It is assumed that the communication refers to a previously established session between the Authenticator and the Server. The Server implicitly decrypts the obtained message with the public key (`pK(`A_k`)`) of the Authenticator. If the value of the received `otp` is as expected by the Server, the actions following this list will be executed.

$Actions_k^{out}(\mathbf{opid} \gg_\gamma \tau^{(?)}\mathbf{[F]} \gg_i \mathbf{otp})$ **where** $O \in F$**.** In this case, an Authenticator, receiving an `opid` as input, generates an `otp` that is sent to the Endpoint (via IPC) in order to be forwarded to the Server. This list of actions starts with the Authenticator (A_k) generating a the `otp (crypt( inv( pK(`A_k`)), opid(Operation,UniqueID)))` and sending it to the Endpoint (E). The Endpoint sends the `otp` to the Server for verification. The Server implicitly decrypts the obtained message with the public key (`pK(`A_k`)`) of the Authenticator. If the value of the received `otp` is as expected by the Server, the actions following this list will be executed.

$Actions_k^{out}(\mathbf{opid} \gg_\gamma \tau^{(?)}\mathbf{[F]} \gg_i \mathbf{otp})$ **where** $O \notin F$**.** This case refers to an authenticator that generates an `otp` (depending from an `opid`), transmits it via IPC or USB interface, and does not attest an Ownership factor. Except for the first action, in which the channel between the Authenticator and the Endpoint is not confidential, the set of actions is identical w.r.t. the case in which $O \in F$.

$Actions_k^{out}(\mathbf{opid} \gg_\gamma \tau^{(?)}\mathbf{[F]} \gg_m \mathbf{otp})$**.** In this case, an Authenticator, receiving an `opid` as input, generates an `otp` that is sent to the Server leveraging the mobile network. The first action, represents the Authenticator (A_k) generating the `otp` and sending it to the Telephony Server. The Telephony Server transmits the received `otp` to the Server, together with the identity of the sender (A_k). The Server implicitly decrypts the obtained message with the public key (`pk(`A_k`)`) of the Authenticator. If the value of the received `otp` (for the specific authenticator A_k) is as expected by the Server, the actions following this list will be executed.

$Actions_k^{out}(\mathbf{otp} \gg_\gamma \tau\mathbf{[F]} \gg_m \mathbf{otp})$**.** In this case, an Authenticator, receiving an `otp` as input, sends the same `otp` to the Server leveraging the mobile network. Such behavior is typical of an

Out-of-band authenticator. In this case, the Authenticator sends the otp to the Telephony Server. At this point, the Telephony Server transmits the received otp to the Server, together with the identity of the sender (A_k). If the value of the received otp (for the specific authenticator A_k) is as expected by the Server, the actions following this list will be executed.

$Actions_k^{out}$(**otp**$\gg_\gamma \tau$**[F]** $\gg_n$ **otp**)**.** In this case, an Authenticator, receiving an otp as input, sends the same otp to the Server via a network connection. As for the previous case, such behavior is typical of an Out-of-band authenticator. In this case, the Authenticator sends the otp directly to the Server. If the value of the received otp (for the specific authenticator A_k) is as expected by the Server, the actions following this list will be executed.

$Actions_k^{out}$(**otp**$\gg_\gamma \tau^{?}$**[F]** $\gg_m$ **otp**)**.** In this case, an Authenticator, receiving an otp as input and showing information regarding the ongoing operation to the user, sends an otp to the Server via the telephony network.

In this case, the Authenticator had previously received an otp and showed Operation to the user. The first step consists in decrypting the received otp (see Table 3.2 for more details), signing it with the private key and sending it to the Telephony Server. At this point, the Telephony Server transmits the received otp to the Server, together with the identity of the sender (A_k). If the value of the received otp (for the specific authenticator A_k) is as expected by the Server, the actions following this list will be executed.

Example 5. *Let us consider the protocol Nordea1. As presented in Section 3.2, it can be represented in SLaMP as* 🔑; opid$\gg_h$ 🖩*[O,K]* $\gg_h$ otp *(formula* (3.2)*). Indicating with* α_{N1} *the result of Actions*(*Nordea1*)*, we calculate it as*

$$\alpha_{N1} = Actions(🔑;\ \texttt{opid} \gg_h 🖩[\mathrm{O,K}]\ \gg_h \texttt{otp})$$

More specifically, we can write

$$\alpha_{N1} = PRE \cdot Actions_1(🔑) \cdot Actions_2(\texttt{opid} \gg_h 🖩[\mathrm{O,K}]\ \gg_h \texttt{otp})$$

Now we compute $Actions_1(🔑)$ *as in Table 3.3. In particular, since* A_1 *is a knowledge factor, it is modeled with the mapping* mem*. Since it is shared between the Server and the User, and it is the first authenticator, we write it as* mem(S,U,1)*.*

$$Actions_1(🔑) = \begin{cases} \mathtt{U}\bullet \rightarrow \bullet\mathtt{E} : \mathtt{mem}(\mathtt{S},\mathtt{U},1) \\ \mathtt{E} \rightarrow \bullet\mathtt{S} : \mathtt{mem}(\mathtt{S},\mathtt{U},1) \end{cases}$$

$Actions_2(\texttt{opid} \gg_h 🖩$*[O,K]* $\gg_h \texttt{otp})$ *requires more attention. As a matter of fact, as defined in Section 3.3.4.1*

```
1   U: Msg Operation
2   U →• E: Operation
3   E →• S: Operation
4   U →• E: mem(S,U,1)
5   E →• S: mem(S,U,1)
6   S : Number UniqueID
7   S •→• E: scrypt(dK(S,A2),opid(Operation,UniqueID))
8   E •→• U: scrypt(dK(S,A2),opid(Operation,UniqueID))
9   U •→• A2: scrypt(dK(S,A2),opid(Operation,UniqueID)))
10  U •→• A2: kF(A2)
11  A2 •→• U: crypt(inv(pK(A2)),opid(Operation,UniqueID))
12  U →• E: crypt(inv(pK(A2)),opid(Operation,UniqueID))
13  E →• S: crypt(inv(pK(A2)),opid(Operation,UniqueID))
```

Listing 3.4: Full structure of α_{N1} from Example 5.

$$Actions_2(A) = Actions_2^{in}(A) \cdot Actions_2^{mid}(A) \cdot Actions_2^{out}(A)$$

where $A = \texttt{opid} \gg_h$ 🖩*[O,K]* $\gg_h \texttt{otp}$.

In the following, we compute $Actions_2^{in}(A)$, $Actions_2^{mid}(A)$ *and* $Actions_2^{out}$ *as in Tables 3.2, 3.3 and 3.4, respectively.*

$$Actions_2^{in}(A) = \begin{cases} \mathtt{S} : \mathtt{Number\ UniqueID} \\ \mathtt{S}\bullet \rightarrow \bullet\mathtt{E} : \mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A}_2),\mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \\ \mathtt{E}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A}_2),\mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \\ \mathtt{U}\bullet \rightarrow \bullet\mathtt{A}_2 : \mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A}_2),\mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \end{cases}$$

$$Actions_2^{mid}(A) = \mathtt{U}\bullet \rightarrow \bullet\mathtt{A}_2 : \mathtt{fK}(\mathtt{U},\mathtt{A}_2)$$

$$Actions_2^{out}(A) = \begin{cases} \mathtt{A}_2\bullet \rightarrow \bullet\mathtt{U} : \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A}_2)),\mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \\ \mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A}_2)),\mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \\ \mathtt{E} \rightarrow \bullet\mathtt{S} : \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A}_2)),\mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \end{cases}$$

The full structure of α_{N1} *is presented in Listing 3.4* □

3.3.4.2 Agents Function

The function $Agents(\alpha)$, returns the set of agents appearing in α. The agents that can appear in an SPS specification generated from a *n*-authenticators SLaMP specification belong to the finite set $\Lambda = \{\mathtt{U},\mathtt{E},\mathtt{S},\mathtt{T},\mathtt{A}_1,\ldots,\mathtt{A}_n\}$, where $\mathtt{A}_k \notin \{🔑\}$. We denote a generic element of Λ with β,β'. The *Agents* function is inductively defined as follows:

$$
\begin{array}{ll}
Agents([\,]) & = \emptyset \\
Agents(\beta \to \beta' : m \cdot \alpha) & = \{\beta, \beta'\} \cup Agents(\alpha) \\
Agents(\beta : d \cdot \alpha) & = \{\beta\} \cup Agents(\alpha)
\end{array}
$$

For each action in α, if the action is empty, the empty set is given. If the action is specified as $\beta \to \beta' : m$, the two roles β and β' are united to the set. Finally, if the action is specified as $\beta : d$, where d is a placeholder for the creation of a fresh variable, the role β is united to the set.

Example 6. *Let us consider the list of actions α_{N1} for Nordea1, presented in Listing 3.4. For brevity, we use α^h to denote the sublist obtained from α by removing the first h elements. Let us calculate $Agents(\alpha_{N1})$.*

$Agents(\alpha_{N1}) = Agents(\mathtt{U} : \mathtt{Msg}\ \mathtt{Operation} \cdot \alpha^1_{N1}) = \{\mathtt{U}\} \cup Agents(\alpha^1_{N1}) =$

$= \{\mathtt{U}\} \cup Agents(\mathtt{U} \to \bullet\mathtt{E} : \mathtt{Operation} \cdot \alpha^2_{N1}) = \{\mathtt{U}\} \cup \{\mathtt{E}\} \cup Agents(\alpha^2_{N1}) =$

$= \{\mathtt{U},\mathtt{E}\} \cup Agents(\mathtt{E} \to \bullet\mathtt{S} : \mathtt{Operation} \cdot \alpha^3_{N1}) = \{\mathtt{U},\mathtt{E}\} \cup \{\mathtt{S}\} \cup Agents(\alpha^3_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{S}\} \cup Agents(\mathtt{U} \to \bullet\mathtt{E} : \mathtt{mem}(\mathtt{S},\mathtt{U},1) \cdot \alpha^4_{N1}) = \{\mathtt{U},\mathtt{E},\mathtt{S}\} \cup Agents(\alpha^4_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{S}\} \cup Agents(\mathtt{E} \to \bullet\mathtt{S} : \mathtt{mem}(\mathtt{S},\mathtt{U},1) \cdot \alpha^5_{N1}) = \{\mathtt{U},\mathtt{E},\mathtt{S}\} \cup Agents(\alpha^5_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{S}\} \cup Agents(\mathtt{S} : \mathtt{NumberUniqueID} \cdot \alpha^6_{N1}) = \{\mathtt{U},\mathtt{E},\mathtt{S}\} \cup Agents(\alpha^6_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{S}\} \cup Agents(\mathtt{S}\bullet \to \bullet\mathtt{E} : \mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A}_2), \mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \cdot \alpha^7_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{S}\} \cup Agents(\alpha^7_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{S}\} \cup Agents(\mathtt{E}\bullet \to \bullet\mathtt{U} : \mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A}_2), \mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \cdot \alpha^8_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{S}\} \cup Agents(\alpha^8_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{S}\} \cup Agents(\mathtt{U}\bullet \to \bullet\mathtt{A}_2 : \mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A}_2), \mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \cdot \alpha^9_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{S}\} \cup \{\mathtt{A}_2\} \cup Agents(\alpha^9_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{S},\mathtt{A}_2\} \cup Agents(\mathtt{U}\bullet \to \bullet\mathtt{A}_2 : \mathtt{fK}(\mathtt{U},\mathtt{A}_2) \cdot \alpha^{10}_{N1}) = \{\mathtt{U},\mathtt{E},\mathtt{S},\mathtt{A}_2\} \cup Agents(\alpha^{10}_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{S},\mathtt{A}_2\} \cup Agents(\mathtt{A}_2\bullet \to \bullet\mathtt{U} : \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A}_2)), \mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \cdot \alpha^{11}_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{S},\mathtt{A}_2\} \cup Agents(\alpha^{11}_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{S},\mathtt{A}_2\} \cup Agents(\mathtt{U} \to \bullet\mathtt{E} : \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A}_2)), \mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \cdot \alpha^{12}_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{S},\mathtt{A}_2\} \cup Agents(\alpha^{12}_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{S},\mathtt{A}_2\} \cup Agents(\mathtt{E} \to \bullet\mathtt{S} : \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A}_2)), \mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID}))) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{S},\mathtt{A}_2\}$

□

Table 3.5: Set of mappings that can appear in a generated SPS specification.

Element	Type	Description
`mem`	`Agent,Agent -> Msg`	the memorized secret established between the User and the Server.
`lus`	`Agent,Agent,Number -> Number`	the mapping implementing the Look-up secret
`seed`	`Agent,Agent -> Number`	the seed that is shared between an Authenticator and the Server that must be used for generating an `otp` which does not depend from `opid`
`pK`	`Agent -> PublicKey`	models the public key associated to an agent
`inv`	`PublicKey -> PrivateKey`	the private key associated to a given public key
`dk`	`Agent, Agent -> SymmetricKey`	the symmetric key (shared between the Server and the Authenticator) for decoding the `opid`.
`fK`	`Agent, Agent -> SymmetricKey`	the Knowledge factor that the user must provide for unlocking the Authenticator.
`fI`	`Agent, Agent -> SymmetricKey`	the Inherence factor that the user must provide for unlocking the Authenticator.

Similarly to Section 3.2.4, here we define a utility function

$$Platform : \Lambda \rightarrow \{Browser, App, Desktop, Mobile, Hardware, Paper, \perp\}$$

In words, $Platform(\beta) = e$ means that the agent β is executed inside e. Moreover, we require $Platform(E) \in \{Browser, App\}$, $Platform(U) = Platform(S) = Platform(T) = \perp$, $Platform(A_k) \in \{Desktop, Mobile, Hardware\}$. For instance, $Platform(A) = Mobile$ means that authenticator A is a software running on a mobile device.

3.3.4.3 Mappings Function

The function $Mappings(\alpha)$ returns the set of mappings appearing in α. To define $Mappings$, we need to introduce some preliminary definitions.

First, we introduce Γ as the (finite) set of names of mappings that can appear in an SPS specification. In symbols $\Gamma = \{\texttt{mem}, \texttt{lus}, \texttt{seed}, \texttt{pK}, \texttt{inv}, \texttt{dk}, \texttt{fK}, \texttt{fI}\}$. An extended description of each mapping is given in Table 3.5.

The utility function $TypeOf$, taking an element of Γ as a parameter, gives its type as result. For example, $TypeOf(mem)$ would give `Number -> Msg` as a result.

The $Mappings$ function is inductively defined as follows.

$$
\begin{array}{ll}
\mathit{Mappings}([\,]) & = \emptyset \\
\mathit{Mappings}(\beta \to \beta' : m \cdot \alpha) & = \mathit{Mappings}(m) \cup \mathit{Mappings}(\alpha) \\
\mathit{Mappings}(\beta : d \cdot \alpha) & = \mathit{Mappings}(\alpha)
\end{array}
$$

Slightly abusing the notation, we apply *Mappings* to a message m as

$$
\begin{array}{ll}
\mathit{Mappings}(f(m_1,\ldots,m_k)) & = \Gamma(f) \cup \bigcup_{i\in\{1,\ldots,k\}} \mathit{Mappings}(m_i) \\
\mathit{Mappings}(v) & = \emptyset
\end{array}
$$

where v is either a variable or a constant and $\Gamma(\bullet)$ is a shorthand for the function

$$
\lambda f. \begin{cases} \{f\} & \text{if } f \in \Gamma \\ \emptyset & \text{otherwise} \end{cases}
$$

In words, for each action in α in the form $\beta \to \beta' : m$, the content of m is matched on the set Γ. If any part of m corresponds to an element in Γ, the corresponding value is united with the set of Mappings.

Example 7. *Let us consider again α_{N1} (the list of actions of Nordea1). The result of $\mathit{Mappings}(\alpha_{N1})$ can be calculated as follows. As for the previous example, we use α^h to denote the sublist obtained from α by removing the first h elements.*

$\mathit{Mappings}(\alpha_{N1}) = \mathit{Mappings}(\mathtt{U} : \mathtt{Msg\ Operation} \cdot \alpha^1_{N1}) = \emptyset \cup \mathit{Mappings}(\alpha^1_{N1}) =$

$= \mathit{Mappings}(\mathtt{U} \to \bullet\mathtt{E} : \mathtt{Operation} \cdot \alpha^2_{N1}) = \mathit{Mappings}(\alpha^2_{N1}) =$

$= \mathit{Mappings}(\mathtt{E} \to \bullet\mathtt{S} : \mathtt{Operation} \cdot \alpha^3_{N1}) = \mathit{Mappings}(\alpha^3_{N1}) =$

$= \mathit{Mappings}(\mathtt{U} \to \bullet\mathtt{E} : \mathtt{mem}(\mathtt{S},\mathtt{U},\mathtt{1}) \cdot \alpha^4_{N1}) = \{\mathtt{mem}\} \cup \mathit{Mappings}(\alpha^4_{N1}) =$

$= \{\mathtt{mem}\} \cup \mathit{Mappings}(\mathtt{E} \to \bullet\mathtt{S} : \mathtt{mem}(\mathtt{S},\mathtt{U},\mathtt{1}) \cdot \alpha^5_{N1}) = \{\mathtt{mem}\} \cup \mathit{Mappings}(\alpha^5_{N1}) =$

$= \{\mathtt{mem}\} \cup \mathit{Mappings}(\mathtt{S} : \mathtt{NumberUniqueID} \cdot \alpha^6_{N1}) = \{\mathtt{mem}\} \cup \mathit{Mappings}(\alpha^6_{N1}) =$

$= \{\mathtt{mem}\} \cup \mathit{Mappings}(\mathtt{S}\bullet \to \bullet\mathtt{E} : \mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A_2}),\mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \cdot \alpha^7_{N1}) =$

$= \{\mathtt{mem}\} \cup \{\mathtt{dK}\} \cup \mathit{Mappings}(\alpha^7_{N1}) =$

$= \{\mathtt{mem},\mathtt{dK}\} \cup \mathit{Mappings}(\mathtt{E}\bullet \to \bullet\mathtt{U} : \mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A_2}),\mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \cdot \alpha^8_{N1}) =$

$= \{\mathtt{mem},\mathtt{dK}\} \cup \mathit{Mappings}(\alpha^8_{N1}) =$

$= \{\mathtt{mem},\mathtt{dK}\} \cup \mathit{Mappings}(\mathtt{U}\bullet \to \bullet\mathtt{A_2} : \mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A_2}),\mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \cdot \alpha^9_{N1}) =$

$= \{\mathtt{mem},\mathtt{dK}\} \cup \mathit{Mappings}(\alpha^9_{N1}) =$

$= \{\mathtt{mem},\mathtt{dK}\} \cup \mathit{Mappings}(\mathtt{U}\bullet \to \bullet\mathtt{A_2} : \mathtt{fK}(\mathtt{U},\mathtt{A_2}) \cdot \alpha^{10}_{N1}) = \{\mathtt{mem},\mathtt{dK}\} \cup \{\mathtt{fK}\} \cup \mathit{Mappings}(\alpha^{10}_{N1}) =$

$= \{\mathtt{mem},\mathtt{dK},\mathtt{fK}\} \cup \mathit{Mappings}(\mathtt{A_2}\bullet \to \bullet\mathtt{U} : \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A_2})),\mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \cdot \alpha^{11}_{N1}) =$

$= \{\mathtt{mem},\mathtt{dK},\mathtt{fK}\} \cup \{\mathtt{pK},\mathtt{inv}\} \cup \mathit{Mappings}(\alpha^{11}_{N1}) =$

Table 3.6: Set of formats that can appear in a generated SPS specification.

Element	Type	Description
`opid`	`(Msg, Number)`	the format of the opid which is transmitted by Server to the Authenticator.
`pair`	`(Agent, Msg)`	the structure containing an otp which is sent to the Telephony Server

$= \{\mathtt{mem}, \mathtt{dK}, \mathtt{fK}, \mathtt{pK}, \mathtt{inv}\} \cup \mathit{Mappings}(\mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A}_2)), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID})) \cdot \alpha_{N1}^{12}) =$

$= \{\mathtt{mem}, \mathtt{dK}, \mathtt{fK}, \mathtt{pK}, \mathtt{inv}\} \cup \mathit{Mappings}(\alpha_{N1}^{12}) =$

$= \{\mathtt{mem}, \mathtt{dK}, \mathtt{fK}, \mathtt{pK}, \mathtt{inv}\} \cup \mathit{Mappings}(\mathtt{E} \rightarrow \bullet\mathtt{S} : \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A}_2)), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID}))) =$

$= \{\mathtt{mem}, \mathtt{dK}, \mathtt{fK}, \mathtt{pK}, \mathtt{inv}\}$

□

3.3.4.4 Formats Function

The function $\mathit{Formats}(\alpha)$ returns the set of formats appearing in α. As for the set of mappings, we need to introduce some preliminary definitions.

We introduce Φ as the (finite) set of names of formats that can appear in an SPS specification. In symbols $\Phi = \{\mathtt{opid}, \mathtt{pair}\}$. An extended description of each format is given in Table 3.6.

As for the case of *Mappings*, we define the function TypeOf, taking an element of Φ as a parameter, gives its type as result. For example, $\mathit{TypeOf}(otp)$ would give `Msg` as a result.

The *Formats* function is inductively defined as follows.

$$\begin{array}{ll} \mathit{Formats}([\,]) & = \emptyset \\ \mathit{Formats}(\beta \rightarrow \beta' : m \cdot \alpha) & = \mathit{Formats}(m) \cup \mathit{Formats}(\alpha) \\ \mathit{Formats}(\beta : d \cdot \alpha) & = \mathit{Formats}(\alpha) \end{array}$$

Slightly abusing the notation, we apply *Formats* to a message m as:

$$\begin{array}{ll} \mathit{Formats}(f(m_1, \ldots, m_k)) & = \Phi(f) \cup \bigcup_{i \in \{1,\ldots,k\}} \mathit{Formats}(m_i) \\ \mathit{Formats}(v) & = \emptyset \end{array}$$

where v is a generic variable or a constant and $\Phi(\bullet)$ is a shorthand for the function

$$\lambda f. \begin{cases} \{f\} & \text{if } f \in \Phi \\ \emptyset & \text{otherwise} \end{cases}$$

As for Γ in the *Mappings* function, for each action in α in the form $\beta \rightarrow \beta' : m$, the content of

m is matched on the set Φ. If any part of *m* corresponds to an element in Φ, the corresponding value is united with the set of Formats.

Example 8. *As for the Mappings function, let us consider again* α_{N1} *(the list of actions of Nordea1) and calculate the result of* $Formats(\alpha_{N1})$.

$Formats(\alpha_{N1}) = Formats(\mathtt{U} : \mathtt{Msg}\ \mathtt{Operation} \cdot \alpha^1_{N1}) = Formats(\alpha^1_{N1}) =$

$= Formats(\mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{Operation} \cdot \alpha^2_{N1}) = Formats(\alpha^2_{N1}) =$

$= Formats(\mathtt{E} \rightarrow \bullet\mathtt{S} : \mathtt{Operation} \cdot \alpha^3_{N1}) = Formats(\alpha^3_{N1}) =$

$= Formats(\mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{mem}(\mathtt{S},\mathtt{U},1) \cdot \alpha^4_{N1}) = Formats(\alpha^4_{N1}) =$

$= Formats(\mathtt{E} \rightarrow \bullet\mathtt{S} : \mathtt{mem}(\mathtt{S},\mathtt{U},1) \cdot \alpha^5_{N1}) = Formats(\alpha^5_{N1}) =$

$= Formats(\mathtt{S} : \mathtt{NumberUniqueID} \cdot \alpha^6_{N1}) = Formats(\alpha^6_{N1}) =$

$= Formats(\mathtt{S}\bullet \rightarrow \bullet\mathtt{E} : \mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A}_2), \mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \cdot \alpha^7_{N1}) =$

$= \{\mathtt{opid}\} \cup Formats(\alpha^7_{N1}) =$

$= \{\mathtt{opid}\} \cup Formats(\mathtt{E}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A}_2), \mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \cdot \alpha^8_{N1}) =$

$= \{\mathtt{opid}\} \cup Formats(\alpha^8_{N1}) =$

$= \{\mathtt{opid}\} \cup Formats(\mathtt{U}\bullet \rightarrow \bullet\mathtt{A}_2 : \mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A}_2), \mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \cdot \alpha^9_{N1}) =$

$= \{\mathtt{opid}\} \cup Formats(\alpha^9_{N1}) =$

$= \{\mathtt{opid}\} \cup Formats(\mathtt{U}\bullet \rightarrow \bullet\mathtt{A}_2 : \mathtt{fK}(\mathtt{U},\mathtt{A}_2) \cdot \alpha^{10}_{N1}) = \{\mathtt{opid}\} \cup Formats(\alpha^{10}_{N1}) =$

$= \{\mathtt{opid}\} \cup Formats(\mathtt{A}_2\bullet \rightarrow \bullet\mathtt{U} : \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A}_2)), \mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \cdot \alpha^{11}_{N1}) =$

$= \{\mathtt{opid}\} \cup Formats(\alpha^{11}_{N1}) =$

$= \{\mathtt{opid}\} \cup Formats(\mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A}_2)), \mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \cdot \alpha^{12}_{N1}) =$

$= \{\mathtt{opid}\} \cup Formats(\alpha^{12}_{N1}) =$

$= \{\mathtt{opid}\} \cup Formats(\mathtt{E} \rightarrow \bullet\mathtt{S} : \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A}_2)), \mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID}))) =$

$= \{\mathtt{opid}\}$

□

3.3.4.5 Knowledge Function

The *Knowledge* function gives the list of specifications representing the initial knowledge of each agent participating to the protocol. In our framework, we assume that an MFA protocol is executed after the preliminary phases (enrollment and binding procedures) have been correctly completed. Therefore, we assume that all the secrets and the authentication factors have been

properly distributed and stored. Therefore, they are part of the initial knowledge of the respective roles and they are unknown to the intruder.

Similarly to Section 3.3.4.3, we need to introduce some preliminary notions.

For each $\beta \in \Lambda$ we define $\Gamma_\beta \subseteq \Gamma$ as the set of names of mappings that can appear in the knowledge of β. We define them as follows.

$$\begin{array}{ll} \Gamma_U = \{\texttt{mem}, \texttt{pK}, \texttt{fK}, \texttt{fI}\} & \Gamma_S = \{\texttt{mem}, \texttt{pK}, \texttt{lus}, \texttt{seed}, \texttt{dK}\} \\ \Gamma_E = \Gamma_T = \{\texttt{pK}\} & \Gamma_{A_k} = \{\texttt{pK}, \texttt{lus}, \texttt{seed}, \texttt{dK}, \texttt{inv}, \texttt{fK}, \texttt{fI}\} \end{array}$$

In words, besides the other agents she interacts with, the User knows the memorized secrets, any public key and the knowledge and the inherence factors that are required to unlock multi-factor authenticators. The Server knows all the memorized and look-up secrets, the public keys, all the seeds that are used for `otp` generation and all the decoding keys for transmitting the `opid` (that have been previuosly shared with the authenticators). The Endpoint and the Telephony server know all the public keys. Finally, an Authenticator knows all the public keys, the knowledge for retrieving a look-up secret, the knowledge and inherence factors that may be needed for unlocking it and, depending on its functionality, either the decoding key for handling the `opid` and its private key (for performing asymmetric encryption of an `otp`), or the seed for generating an `otp` (not related to an `opid`).

Moreover, we define the knowledge environment κ (being $\perp$ the empty environment) that binds an agent β to its initial knowledge. The initial knowledge is a set that can contain both agents and instances of mappings. We use $\kappa[\beta \mapsto K]$ to denote the mapping that binds β to K and behaves as κ for any other β'. Also, we write $[\beta \mapsto K]$ as shorthand for $\perp[\beta \mapsto K]$. Furthermore, we introduce the operator $\oplus$ to compose two knowledge environments as $(\kappa \oplus \kappa')(\beta) = \kappa(\beta) \cup \kappa'(\beta)$.

Now, we inductively define the function $Knowledge_\beta(\alpha)$, for each agent $\beta \in \Lambda$, as

$$\begin{array}{ll} Knowledge_\beta([\,]) = Knowledge_\beta(\beta : v) = Knowledge_\beta(\beta' : v) = \perp & \\ Knowledge_\beta(\beta \to \beta' : m \cdot \alpha) & = Knowledge_\beta(\alpha) \oplus Knowledge_\beta(m) \oplus [\beta \mapsto \{\beta'\}] \\ Knowledge_\beta(\beta' \to \beta : m \cdot \alpha) & = Knowledge_\beta(\alpha) \oplus Knowledge_\beta(m) \oplus [\beta \mapsto \{\beta'\}] \\ Knowledge_\beta(\beta' \to \beta'' : m \cdot \alpha) & = Knowledge_\beta(\alpha) \end{array}$$

Slightly abusing the notation, we apply $Knowledge_\beta$ to a message m. In particular, with $\beta \in \{U, E, S, T\}$ we define $Knowledge_\beta(m)$ as

$$Knowledge_\beta(f(m_1, \ldots, m_k)) = \bigcup_i Knowledge_\beta(m_i) \cup \begin{cases} \{f(m_1, \ldots, m_k)\} \text{ if } f \in \Gamma_\beta \\ \emptyset \text{ otherwise} \end{cases}$$

Instead, when $\beta \in \{A_1, \ldots, A_n\} \in \Lambda$ we define $Knowledge_\beta$ as

$$Knowledge_\beta(f(m_1,\ldots,m_k)) = \bigcup_i Knowledge_\beta(m_i) \cup \begin{cases} \{f(m_1,\ldots,m_k)\} \text{ if } f \in \Gamma_\beta \text{ and } \beta \text{ appears in some } m_i \\ \emptyset \text{ otherwise} \end{cases}$$

Thus, $Knowledge(\alpha)$ is defined as the minimal mapping κ s.t. $\kappa(\beta) = Knowledge_\beta(\alpha)$.

Example 9. *We now calculate* $Knowledge(\alpha_{N1})$*, where* α_{N1} *is defined as in Example 5. This requires us to calculate* $Knowledge_U(\alpha_{N1})$*,* $Knowledge_E(\alpha_{N1})$*,* $Knowledge_S(\alpha_{N1})$ *and* $Knowledge_{A_2}(\alpha_{N1})$ *as follows.*

$Knowledge_U(\alpha_{N1}) = Knowledge_U(\mathtt{U} : \mathtt{Msg}\ \mathtt{Operation} \cdot \alpha^1_{N1}) = \{\mathtt{U}\} \cup Knowledge_U(\alpha^1_{N1}) =$

$= \{\mathtt{U}\} \cup Knowledge_U(\mathtt{U} \to \bullet\mathtt{E} : \mathtt{Operation} \cdot \alpha^2_{N1}) = \{\mathtt{U}\} \cup \{\mathtt{E}\} \cup Knowledge_U(\alpha^2_{N1}) =$

$= \{\mathtt{U},\mathtt{E}\} \cup Knowledge_U(\mathtt{E} \to \bullet\mathtt{S} : \mathtt{Operation} \cdot \alpha^3_{N1}) = \{\mathtt{U},\mathtt{E}\} \cup Knowledge_U(\alpha^3_{N1}) =$

$= \{\mathtt{U},\mathtt{E}\} \cup Knowledge_U(\mathtt{U} \to \bullet\mathtt{E} : \mathtt{mem}(\mathtt{S},\mathtt{U},1) \cdot \alpha^4_{N1}) = \{\mathtt{U},\mathtt{E}\} \cup \{\mathtt{mem}(\mathtt{S},\mathtt{U},1)\} \cup Knowledge_U(\alpha^4_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{mem}(\mathtt{S},\mathtt{U},1)\} \cup Knowledge_U(\mathtt{E} \to \bullet\mathtt{S} : \mathtt{mem}(\mathtt{S},\mathtt{U},1) \cdot \alpha^5_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{mem}(\mathtt{S},\mathtt{U},1)\} \cup Knowledge_U(\alpha^5_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{mem}(\mathtt{S},\mathtt{U},1)\} \cup Knowledge_U(\mathtt{S} : \mathtt{NumberUniqueID} \cdot \alpha^6_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{mem}(\mathtt{S},\mathtt{U},1)\} \cup Knowledge_U(\alpha^6_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{mem}(\mathtt{S},\mathtt{U},1)\} \cup Knowledge_U(\mathtt{S}\bullet \to \bullet\mathtt{E} : \mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A_2}),\mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \cdot \alpha^7_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{mem}(\mathtt{S},\mathtt{U},1)\} \cup Knowledge_U(\alpha^7_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{mem}(\mathtt{S},\mathtt{U},1)\} \cup Knowledge_U(\mathtt{E}\bullet \to \bullet\mathtt{U} : \mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A_2}),\mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \cdot \alpha^8_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{mem}(\mathtt{S},\mathtt{U},1)\} \cup \{\mathtt{dK}(\mathtt{S},\mathtt{A_2})\} \cup Knowledge_U(\alpha^8_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{mem}(\mathtt{S},\mathtt{U},1),\mathtt{dK}(\mathtt{S},\mathtt{A_2})\}$

$\quad \cup Knowledge_U(\mathtt{U}\bullet \to \bullet\mathtt{A_2} : \mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A_2}),\mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \cdot \alpha^9_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{mem}(\mathtt{S},\mathtt{U},1),\mathtt{dK}(\mathtt{S},\mathtt{A_2})\} \cup \{\mathtt{A_2}\} \cup Knowledge_U(\alpha^9_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{A_2},\mathtt{mem}(\mathtt{S},\mathtt{U},1),\mathtt{dK}(\mathtt{S},\mathtt{A_2})\} \cup Knowledge_U(\mathtt{U}\bullet \to \bullet\mathtt{A_2} : \mathtt{fK}(\mathtt{U},\mathtt{A_2}) \cdot \alpha^{10}_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{A_2},\mathtt{mem}(\mathtt{S},\mathtt{U},1),\mathtt{dK}(\mathtt{S},\mathtt{A_2})\} \cup \{\mathtt{fK}(\mathtt{U},\mathtt{A_2})\} \cup Knowledge_U(\alpha^{10}_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{A_2},\mathtt{mem}(\mathtt{S},\mathtt{U},1),\mathtt{dK}(\mathtt{S},\mathtt{A_2}),\mathtt{fK}(\mathtt{U},\mathtt{A_2})\}$

$\quad \cup Knowledge_U(\mathtt{A_2}\bullet \to \bullet\mathtt{U} : \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A_2})),\mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \cdot \alpha^{11}_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{A_2},\mathtt{mem}(\mathtt{S},\mathtt{U},1),\mathtt{dK}(\mathtt{S},\mathtt{A_2}),\mathtt{fK}(\mathtt{U},\mathtt{A_2})\} \cup Knowledge_U(\alpha^{11}_{N1}) =$

$= \{\mathtt{U},\mathtt{E},\mathtt{A_2},\mathtt{mem}(\mathtt{S},\mathtt{U},1),\mathtt{dK}(\mathtt{S},\mathtt{A_2}),\mathtt{fK}(\mathtt{U},\mathtt{A_2})\}$

$\quad \cup Knowledge_U(\mathtt{U} \to \bullet\mathtt{E} : \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A_2})),\mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})) \cdot \alpha^{12}_{N1}) =$

$= \{\mathtt{U}, \mathtt{E}, \mathtt{A}_2, \mathtt{mem}(\mathtt{S},\mathtt{U},1), \mathtt{dK}(\mathtt{S},\mathtt{A}_2), \mathtt{fK}(\mathtt{U},\mathtt{A}_2)\} \cup \mathit{Knowledge}_U(\alpha_{N1}^{12}) =$

$= \{\mathtt{U}, \mathtt{E}, \mathtt{A}_2, \mathtt{mem}(\mathtt{S},\mathtt{U},1), \mathtt{dK}(\mathtt{S},\mathtt{A}_2), \mathtt{fK}(\mathtt{U},\mathtt{A}_2)\}$

$\quad \cup \mathit{Knowledge}_U(\mathtt{E} \rightarrow \bullet\mathtt{S} : \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A}_2)), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID}))) =$

$= \{\mathtt{U}, \mathtt{E}, \mathtt{A}_2, \mathtt{mem}(\mathtt{S},\mathtt{U},1), \mathtt{dK}(\mathtt{S},\mathtt{A}_2), \mathtt{fK}(\mathtt{U},\mathtt{A}_2)\}$

Similarly to $\mathit{Knowledge}_U(\alpha_{N1})$*, we compute* $\mathit{Knowledge}_E(\alpha_{N1}) = \{\mathtt{U}, \mathtt{E}, \mathtt{S}, \mathtt{pK}(\mathtt{A}_2)\}$ *and* $\mathit{Knowledge}_S(\alpha_{N1}) = \{\mathtt{U}, \mathtt{E}, \mathtt{S}, \mathtt{mem}(\mathtt{S},\mathtt{U},1), \mathtt{pK}(\mathtt{A}_2), \mathtt{dK}(\mathtt{S},\mathtt{A}_2)\}$.
Instead, we compute $\mathit{Knowledge}_{A_2}(\alpha_{N1})$ *as follows.*

$\mathit{Knowledge}_{A_2}(\alpha_{N1}) = \mathit{Knowledge}_{A_2}(\mathtt{U} : \mathtt{MsgOperation} \cdot \alpha_{N1}^{1}) = \mathit{Knowledge}_{A_2}(\alpha_{N1}^{1}) =$

$= \mathit{Knowledge}_{A_2}(\mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{Operation} \cdot \alpha_{N1}^{2}) = \mathit{Knowledge}_{A_2}(\alpha_{N1}^{2}) =$

$= \mathit{Knowledge}_{A_2}(\mathtt{E} \rightarrow \bullet\mathtt{S} : \mathtt{Operation} \cdot \alpha_{N1}^{3}) = \mathit{Knowledge}_{A_2}(\alpha_{N1}^{3}) =$

$= \mathit{Knowledge}_{A_2}(\mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{mem}(\mathtt{S},\mathtt{U},1) \cdot \alpha_{N1}^{4}) = \mathit{Knowledge}_{A_2}(\alpha_{N1}^{4}) =$

$= \mathit{Knowledge}_{A_2}(\mathtt{E} \rightarrow \bullet\mathtt{S} : \mathtt{mem}(\mathtt{S},\mathtt{U},1) \cdot \alpha_{N1}^{5}) = \mathit{Knowledge}_{A_2}(\alpha_{N1}^{5}) =$

$= \mathit{Knowledge}_{A_2}(\mathtt{S} : \mathtt{NumberUniqueID} \cdot \alpha_{N1}^{6}) = \mathit{Knowledge}_{A_2}(\alpha_{N1}^{6}) =$

$= \mathit{Knowledge}_{A_2}(\mathtt{S}\bullet \rightarrow \bullet\mathtt{E} : \mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A}_2), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID})) \cdot \alpha_{N1}^{7}) =$

$= \mathit{Knowledge}_{A_2}(\alpha_{N1}^{7}) =$

$= \mathit{Knowledge}_{A_2}(\mathtt{E}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A}_2), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID})) \cdot \alpha_{N1}^{8}) =$

$= \mathit{Knowledge}_{A_2}(\alpha_{N1}^{8}) =$

$= \mathit{Knowledge}_{A_2}(\mathtt{U}\bullet \rightarrow \bullet\mathtt{A}_2 : \mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A}_2), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID})) \cdot \alpha_{N1}^{9}) =$

$= \{\mathtt{U}, \mathtt{A}_2, \mathtt{dK}(\mathtt{S},\mathtt{A}_2)\} \cup \mathit{Knowledge}_{A_2}(\alpha_{N1}^{9}) =$

$= \{\mathtt{U}, \mathtt{A}_2, \mathtt{dK}(\mathtt{S},\mathtt{A}_2)\} \cup \mathit{Knowledge}_{A_2}(\mathtt{U}\bullet \rightarrow \bullet\mathtt{A}_2 : \mathtt{fK}(\mathtt{U},\mathtt{A}_2) \cdot \alpha_{N1}^{10}) =$

$= \{\mathtt{U}, \mathtt{A}_2, \mathtt{dK}(\mathtt{S},\mathtt{A}_2)\} \cup \{\mathtt{fK}(\mathtt{U},\mathtt{A}_2)\} \cup \mathit{Knowledge}_{A_2}(\alpha_{N1}^{10}) =$

$= \{\mathtt{U}, \mathtt{A}_2, \mathtt{dK}(\mathtt{S},\mathtt{A}_2), \mathtt{fK}(\mathtt{U},\mathtt{A}_2)\}$

$\quad \cup \mathit{Knowledge}_{A_2}(\mathtt{A}_2\bullet \rightarrow \bullet\mathtt{U} : \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A}_2)), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID})) \cdot \alpha_{N1}^{11}) =$

$= \{\mathtt{U}, \mathtt{A}_2, \mathtt{dK}(\mathtt{S},\mathtt{A}_2), \mathtt{fK}(\mathtt{U},\mathtt{A}_2)\} \cup \{\mathtt{pK}(\mathtt{A}_2), \mathtt{inv}(\mathtt{pK}(\mathtt{A}_2))\} \cup \mathit{Knowledge}_{A_2}(\alpha_{N1}^{11}) =$

$= \{\mathtt{U}, \mathtt{A}_2, \mathtt{dK}(\mathtt{S},\mathtt{A}_2), \mathtt{fK}(\mathtt{U},\mathtt{A}_2), \mathtt{pk}(\mathtt{A}_2), \mathtt{inv}(\mathtt{pk}(\mathtt{A}_2))\}$

$\quad \cup \mathit{Knowledge}_{A_2}(\mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A}_2)), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID})) \cdot \alpha_{N1}^{12}) =$

$= \{\mathtt{U}, \mathtt{A}_2, \mathtt{dK}(\mathtt{S},\mathtt{A}_2), \mathtt{fK}(\mathtt{U},\mathtt{A}_2), \mathtt{pk}(\mathtt{A}_2), \mathtt{inv}(\mathtt{pk}(\mathtt{A}_2))\}$

$\quad \cup \mathit{Knowledge}_{A_2}(\alpha_{N1}^{12}) =$

$= \{\mathtt{U}, \mathtt{A}_2, \mathtt{dK}(\mathtt{S}, \mathtt{A}_2), \mathtt{fK}(\mathtt{U}, \mathtt{A}_2), \mathtt{pk}(\mathtt{A}_2), \mathtt{inv}(\mathtt{pk}(\mathtt{A}_2))\}$

$\cup \mathit{Knowledge}_{A_2}(\mathtt{E} \rightarrow \bullet\mathtt{S} : \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A}_2)), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID}))) =$

$= \{\mathtt{U}, \mathtt{A}_2, \mathtt{dK}(\mathtt{S}, \mathtt{A}_2), \mathtt{fK}(\mathtt{U}, \mathtt{A}_2), \mathtt{pk}(\mathtt{A}_2), \mathtt{inv}(\mathtt{pk}(\mathtt{A}_2))\}$

We conclude by observing that $\mathit{Knowledge}(\alpha_{N1}) = \kappa$ *s.t.*

$\kappa(U) = \{\mathtt{U}, \mathtt{E}, \mathtt{A}_2, \mathtt{mem}(\mathtt{S}, \mathtt{U}, 1), \mathtt{pK}(\mathtt{A}_2), \mathtt{fK}(\mathtt{U}, \mathtt{A}_2)\}$
$\kappa(E) = \{\mathtt{U}, \mathtt{E}, \mathtt{S}, \mathtt{pK}(\mathtt{A}_2)\}$
$\kappa(S) = \{\mathtt{U}, \mathtt{E}, \mathtt{S}, \mathtt{mem}(\mathtt{S}, \mathtt{U}, 1), \mathtt{dK}(\mathtt{S}, \mathtt{A}_2), \mathtt{pk}(\mathtt{A}_2)\}$
$\kappa(A_2) = \{\mathtt{U}, \mathtt{A}_2, \mathtt{dK}(\mathtt{S}, \mathtt{A}_2), \mathtt{fK}(\mathtt{U}, \mathtt{A}_2), \mathtt{pk}(\mathtt{A}_2), \mathtt{inv}(\mathtt{pk}(\mathtt{A}_2))\}$

□

3.3.4.6 Goals

The goal of any authentication protocol is to authenticate the user.

To model this, any SPS specification translated from SLaMP has the following goal

GOAL = `S authenticates U on Operation`

In words, *GOAL* amounts to requiring the Server to authenticate the User while agreeing on the value of `Operation`.

3.3.4.7 Generation of SPS

The specification in SPS of the specified protocol is obtained through the function *Sem*. The definition of *Sem* is given in Listing 3.5.

In Listing 3.5, we introduce some syntactic sugar. In particular, we use α for $\mathit{Actions}(A_1;\ldots;A_n)$ (see Section 3.3.4.1) and $\mathit{Mappings}(\alpha) : \mathit{TypeOf}(\mathit{Mappings}(\alpha))$ (line 5) as a shorthand for $f_1{:}\mathit{TypeOf}(f_1);\ldots;f_n{:}\mathit{TypeOf}(f_n)$, where $\mathit{Mappings}(\alpha) = \{f_1,\ldots,f_n\}$. The same goes for $\mathit{Formats}(\alpha) : \mathit{TypeOf}(\mathit{Formats}(\alpha))$ (line 8). Summing up, the result of *Sem* amounts to the juxtaposition of the fragments returned by the functions previously defined in this section (and *GOAL*).

Example 10. *Let us now generate the SPS specification of Nordea1.*

We can obtain the final SPS by combining the results of $\mathit{Actions}(\mathbf{\Large 🔑}; \mathtt{opid} \gg_h \mathbf{\Large 🖩}[\mathrm{O,K}] \gg_h \mathtt{otp}) = \alpha_{N1}$ *(computed in Example 5),* $\mathit{Agents}(\alpha_{N1})$ *(obtained in Example 6),* $\mathit{Mappings}(\alpha_{N1})$ *(obtained in Example 7),* $\mathit{Formats}(\alpha_{N1})$ *(obtained in Example 8),* $\mathit{Knowledge}(\alpha_{N1})$ *(obtained in Example 9), that we calculated in the previous section. The final result is presented in Listing 3.6.* □

```
1  Types:
2          Agent Agents(α);
3
4  Mappings:
5          Mappings(α) : TypeOf(Mappings(α));
6
7  Formats:
8          Formats(α) : TypeOf(Formats(α));
9
10 Knowledge:
11         Knowledge_U(α);
12         Knowledge_E(α);
13         Knowledge_T(α);
14         Knowledge_S(α);
15         Knowledge_A1(α);
16         ⋮
17         Knowledge_AN(α);
18
19 Actions:
20         α
21
22 Goals:
23         GOAL
```

Listing 3.5: Definition of $Sem(A_1;\ldots;A_n)$.

3.4 Compliance w.r.t. the NIST Classification

In [NIS17] a classification of the authenticators is provided. Such a classification is highly influential and most manufacturers and service providers comply with it. In this section we show that our modeling language is expressive enough to include the definitions given there. Each definition amounts to a constraint over the structure of a generic authenticator as defined in (3.1). The mapping between the definitions and our modeling language is presented in Table 3.7. Below we discuss our encoding (with the exception of 🔑 and ▦ which are straightforward).

3.4.1 Out-of-Band Devices

According to [NIS17, §5.1.3] out-of-band authenticators are physical devices that are uniquely addressable and communicate over a distinct, namely *secondary*, channel (w.r.t. the *primary* channel used by the endpoint). Out-of-band devices include both dedicated and general purpose hardware as far as they satisfy one of the following conditions.

```
1  Types:
2          Agents U, E, S, A2;
3
4  Mappings:
5          mem : Agent,Agent,Number -> Msg;
6          pK: Agent,Agent -> PublicKey;
7          inv: PublicKey -> PrivateKey;
8          fK: Agent,Agent -> SymmetricKey;
9          dK: Agent,Agent -> SymmetricKey;
10
11 Formats:
12         opid(Msg,Number);
13
14 Knowledge:
15         U: U, E, A2, mem(S,U,1), pk(A2), fK(U,A2);
16         E: E, U, S, pK(A2);
17         S: S, U, E, mem(S,U,1), pK(A2), dK(S,A2);
18         A2: U, E, A2, dK(S,A2), fK(U,A2), pk(A2), inv(pK(A2));
19
20 Actions:
21         U: Msg Operation
22         U →• E: Operation
23         E →• S: Operation
24         U →• E: mem(S,U,1)
25         E →• S: mem(S,U,1)
26         S : Number UniqueID
27         S •→• E: scrypt(dK(S,A2),opid(Operation,UniqueID))
28         E •→• U: scrypt(dK(S,A2),opid(Operation,UniqueID))
29         U •→• A2: scrypt(dK(S,A2),opid(Operation,UniqueID)))
30         U •→• A2: fK(U,A2)
31         A2 •→• U: crypt(inv(pK(A2)),opid(Operation,UniqueID))
32         U →• E: crypt(inv(pK(A2)),opid(Operation,UniqueID))
33         E →• S: crypt(inv(pK(A2)),opid(Operation,UniqueID))
34
35 Goals:
36         S authenticates U over Operation
```

Listing 3.6: SPS specification of Nordea1.

1. The secondary channel is used to receive a data item for the user.
2. The secondary channel is used to transmit a data item from the user.
3. The secondary channel is used to send and receive a data item on which the user must agree.

For instance, a common practice is to receive an SMS containing an `otp`. This case complies

with the first condition, i.e., it relies on the mobile telephony network as a secondary channel. Similarly, some MFA protocols require the user to call a secure number from their mobile. This behavior matches the second condition. Finally, the second protocol presented in Section 1.1 is an instance of the third case. As a matter of fact, the user receives a notification on her smartphone. The smartphone connection is a secondary channel w.r.t. the browser connection.

3.4.2 Single and Multi-Factor OTP Device

In [NIS17, §5.1.4 and §5.1.5] these authenticators are defined as devices embedding some seed number used for the generation of OTPs. This category includes both hardware devices and software-based OTP generators installed on devices such as mobile phones. They are distinguished from the out-of-band authenticators as they do not rely on a secondary channel. Moreover, their output can either directly go to the endpoint through an inter-process connection (i) or be copied by the user (h).

3.4.3 Single and Multi-Factor Cryptographic Device

A single or multi-factor cryptographic device [NIS17, §5.1.7 and §5.1.9] is a hardware device that performs some cryptographic operation (e.g., digital signature) on the operation identifier and directly interacts (I/O) with the user endpoint.

3.4.4 Single and Multi-Factor Cryptographic Software

Basically, these authenticators [NIS17, §5.1.6 and §5.1.8] are the software counterparts of the previous category.

3.4.5 Observations

It is worth noticing that our definitions admit intersections between the authenticators. For instance, according to Table 3.7, $\texttt{opid} \gg_i \text{🖩}[\mathrm{O}] \gg_i \texttt{otp}$ is both an OTP and a cryptographic device. This is not actually allowed by the classification of [NIS17]. The reason is that the distinction between these two authenticators is based on an internal feature, i.e., whether they use cryptography or not, that the user cannot observe. As a consequence, our language does not perfecly comply with the categories of [NIS17]. This is expected as, under our working assumptions, we are only interested in classifying authenticators that the user can recognize.

Table 3.7: Mapping NIST definitions to patterns.

Name	Constraints	Example
Memorized Secret	🔑	🔑
Look-up Secret	▦	▦
Out-of-Band Device	$\bigvee \begin{cases} \tau \in \{\text{📱}, \text{📱}^?, \text{💻}, \text{💻}^?\} \wedge \delta \neq \varepsilon \wedge \gamma' = h \\ \tau \in \{\text{📱}, \text{📱}^?, \text{💻}, \text{💻}^?\} \wedge \delta \neq \varepsilon \wedge \gamma \in \{h, o\} \\ \tau \in \{\text{📱}, \text{📱}^?, \text{💻}, \text{💻}^?\} \wedge \delta \neq \varepsilon \wedge \gamma, \gamma' \in \{n, m\} \end{cases}$	otp$\gg_m$ 📱$^?$ $\gg_h$ otp
Single-Factor OTP Device	$\tau \in \{\text{▦}, \text{▦}^?, \text{🔐}, \text{🔐}^?\} \wedge \bar{a} = \{O\} \wedge \gamma \notin \{n, m\} \wedge \gamma' \in \{h, i\}$	opid$\gg_o$ ▦$^?$[O] $\gg_h$ otp
Multi-Factor OTP Device	$\tau \in \{\text{▦}, \text{▦}^?, \text{🔐}, \text{🔐}^?\} \wedge (O \in \bar{a} \wedge \lvert\bar{a}\rvert > 1) \wedge \gamma \notin \{n, m\} \wedge \gamma' \in \{h, i\}$	$\varepsilon \gg_h$ 🔐[O,I] $\gg_h$ otp
Single-Factor Cryptographic Device	$\tau \in \{\text{▦}, \text{▦}^?\} \wedge \bar{a} = \{O\} \wedge \gamma = i \wedge \delta = \texttt{opid} \wedge \gamma' = i$	opid$\gg_i$ ▦$^?$[O] $\gg_i$ otp
Multi-Factor Cryptographic Device	$\tau \in \{\text{▦}, \text{▦}^?\} \wedge (O \in \bar{a} \wedge \lvert\bar{a}\rvert > 1) \wedge \gamma = i \wedge \delta = \texttt{opid} \wedge \gamma' = i$	opid$\gg_i$ ▦[O,K] $\gg_i$ otp
Single-Factor Cryptographic Software	$\tau \in \{\text{🔐}, \text{🔐}^?\} \wedge \bar{a} = \{O\} \wedge \gamma = i \wedge \delta = \texttt{opid} \wedge \gamma' = i$	opid$\gg_i$ 🔐$^?$[O] $\gg_i$ otp
Multi-Factor Cryptographic Software	$\tau \in \{\text{🔐}, \text{🔐}^?\} \wedge (O \in \bar{a} \wedge \lvert\bar{a}\rvert > 1) \wedge \gamma = i \wedge \delta = \texttt{opid} \wedge \gamma' = i$	opid$\gg_i$ 🔐$^?$[O,I] $\gg_i$ otp

3.5 Language for preliminary phases

Besides specifying a protocol design, we propose a language for describing those phases (namely, the *Enrollment* and the *Binding* phases) that need to be performed for allowing the user to execute an MFA protocol.

In particular, we focus on the modalities used for verifying the user identity during enrollment, i.e., identity proofing, and binding phases (the processes for associating the authenticators to the user identity). In the following sections, we provide further details on how Enrollment and Binding phases are modeled.

3.5.1 Enrollment

In this phase users are identified and associated to their digital identity. We distinguish between two possible executions of identity proofing, i.e., in person or remotely. Specifically, we identify two modalities for user identification, namely 🏢 in which users should go to a local office and be identified in person and 🌍 in which users are identified by interacting remotely, e.g., through a web portal or a call service.

3.5.2 Binding

Binding is a procedure that users should perform to associate a new authenticator to their digital identity. As described in Section 2.3.3, we consider it to be consistent of three steps, namely *request*, *delivery* and *activation*. Each of these steps may involve user identification.

Clearly, identity checks during the binding phase can be carried out in the same way as for identity proofing (see above). Nevertheless, since the user has been previously enrolled, she already has a digital identity and, possibly, a previously bound authenticator. This enables one additional identification modalities, i.e., through an MFA protocol. We denote it with . In terms of level of assurance for the identification, we claim that is better than and is better than both. Moreover, a number of service providers allows the binding of one or more authenticators just after the enrollment phase. When this happens, we denote the *request* of these authenticators using symbol **E**.

In our language, the three steps of the binding procedure are be presented as a triple. Each element of the triple either contains one of the aforementioned symbols, i.e., , , , or "–" if the step requires no action (the first element of the triple – *request* – can also contain symbol **E**). For instance, (, ,) indicates a binding procedure in which the user requests an authenticator in person (), receives it through a remote delivery system () and activates it by running an MFA protocol ().

Notice that some services offer alternatives to carry out these steps. When this occurs, we only consider the operation with lower level of assurance. Also, it may happen that two operations are needed to be carried out concurrently, e.g., an activation may require both and . In this case, we only consider the operation with higher level of assurance as an attacker has to compromise both in order to gain control of the authenticator.

Chapter 4

Analysis Framework

In this chapter, the framework for analyzing an MFA protocol is presented. The first section focuses on the evaluation of a protocol in terms of resistance against a set of attacker model. Therefore, we define the attacker models that we consider for the analysis, their application conditions and their counterparts in SPS. Then, a second part on the compliance with regulatory aspects, requirements and best practices is proposed. In particular, we present a set of security requirements against which protocols can be evaluated, along with a set of best practices. Finally, a metric for evaluating the ease-of-use of an MFA protocol is presented.

4.1 Security Analysis

The robustness of MFA protocols against attacks is a critical aspect in the evaluation of their security. In our analysis, we evaluate the security level provided by an MFA protocol by comparing it against a set of attackers.

We firstly define the set of attacker models in terms of capabilities and application conditions on the authenticators specified in SLaMP. Then, we describe how the attackers are modeled by presenting their effects on the SPS specification of a protocol.

4.1.1 Attacker Models

Definition. An attacker is a total function $f : \{Browser, App\} \times (A \cup \{☠\}) \rightarrow A \cup \{☠\}$, where ☠ denotes the unit element w.r.t. ";" (the sequence operator), i.e., $☠;P = P;☠ = P$ for any sequence of authenticators P.

Each attacker model is characterized by an application condition, i.e., a set of capabilities in

terms of which authenticators an attacker can compromise. Intuitively, $f_e(A) = Y$ means that the capabilities of the attacker allow her to treat the authenticator A (appearing in a protocol with endpoint e) as if it was Y, i.e., A and Y are equivalent with respect to f_e. When $f_e(A) = \text{☠}$ we say that attacker f neutralized the authenticator A.[1] Instead, when $f_e(A) = Y$ and $A \neq Y \neq \text{☠}$ we say that f partially compromises A.[2] Finally, when $f_e(A) = A$ we say that f does not affect A. Given an MFA protocol $P = A_1;\ldots;A_n$, s.t. $Endpoint(P) = e$, we define $f_e(P) = f_e(A_1);\ldots;f_e(A_n)$ and we say that f neutralizes P whenever $f_e(P) = \text{☠}$.

The NIST [NIS17] defines several attacker models. Here, we follow the same approach but with few, minor refinements (see below). Briefly, these refinements are necessary to apply the attacker models to our definition of MFA protocol.

Note that, under our assumptions, the applicability of an attacker model to an MFA protocol does not automatically result in an actual threat. In fact, our attacker models represent the set of resources and capabilities that an adversary should have to interact with the elements of an MFA protocol. Reasonably, a threat analysis should consider the applicable attacker models to check whether an attacker can effectively authenticate instead of the user. Such threat analysis is beyond the scope of this work.

Next, we present the attacker models that we consider in our evaluation process. For each of them, we provide a brief explanation in natural language as well as a formal definition (right-hand side). When an authenticator $A_k \in \Omega$ does not appear among the inputs of a function f_e, we intend that $f_e(A_k) = A_k$, i.e., the attacker corresponding to f_e does not affect A_k.

4.1.1.1 Device Thief (DT)

Among all possible threats, one of the most considered - in every application scenario - is the *Device Thief*, i.e., a malicious agent who has the ability to steal a physical object. As a matter of fact, almost every online service implementing MFA offers a procedure in the case that "the device has been stolen". On the other hand, DT models another frequent situation: the device loss. Indeed, in both cases, an unknown person gets the possession of an object which can be used for authentication purposes. In other words, DT targets authenticators relying on ownership factors by stealing them.

The effects of DT on the authenticators that it can target are reported in Table 4.1.

Lines 1, 2 and 4 of Table 4.1. Intuitively, DT can effectively neutralize any single-factor ownership-based authenticator. Indeed, DT can interact with the authenticator in place of the user: as long as it provides the necessary input (as an `opid`, for instance), DT can obtain the authenticator out-

[1] Also we require that $f_e(\text{☠}) = \text{☠}$ for every f and e.

[2] For the sake of presentation, we write $\not{\varphi}$ (with $\varphi \in \{O,K,I\}$) when an attacker compromises a multi-factor authenticator by reducing it to the very same authenticator but for the elimination of φ.

put and send it to the server. However, if an authenticator attests multiple AFs, hence requiring the user to produce a Knowledge or an Inherence factor to unlock the functionality, DT cannot obtain the output (since it cannot provide the requested factors).

For instance, this is the case for ▦: its mere possession would allow the attacker to provide the server with the requested secret.

A	$\mathrm{DT}_e(A)$
1) ▦	☠
2) $\delta \gg_\gamma$ 🖩$^{(?)}$[O] $\gg_\gamma \delta'$	☠
3) $\delta \gg_\gamma$ 🖩$^{(?)}$[$\{O\}\cup$F] $\gg_\gamma \delta'$	$\not\varnothing$
4) $\delta \gg_\gamma$ 🔐$^{(?)}$[O] $\gg_\gamma \delta'$	☠
5) $\delta \gg_\gamma$ 🔐$^{(?)}$[$\{O\}\cup$F] $\gg_\gamma \delta'$	$\not\varnothing$

Table 4.1: Capabilites of DT.

Notice that DT also affects out-of-band authenticators. In particular, DT neutralizes 🖬, i.e., a special case of $\cdot \gg$ 🖩[O] $\gg \cdot$, and applies to 📱, which behaves as 🔐.

Lines 3 and 5 of Table 4.1. DT affects neither knowledge (K) nor inherence (I) factors. Indeed, even if DT can steal the authenticator, it cannot provide the requested factors to unlock it. Thus, authenticator devices and software relying on an ownership factor (together with some other factors) are affected by DT only partially ($\not\varnothing$). For instance, $\mathrm{DT}(\texttt{opid} \gg_h$ 🖩[O,K] $\gg_h \texttt{otp}) = \texttt{opid} \gg_h$ 🖩[K] $\gg_h \texttt{otp}$.

4.1.1.2 Authenticator Duplicator (AD)

We define the *Authenticator Duplicator* as a malicious agent who is able to make a copy of an authenticator without the user's knowledge. In our modeling, the duplication can be of different types. For instance, we consider AD able to attack ▦, by taking a picture to it. On the other hand, AD is also able to duplicate private keys and software pieces.

The latter case seems to be of relevant concern, especially on mobile devices: installing software authenticators on such multi-purpose devices expose them both to the security weaknesses of the devices themselves (as for [R.19]) and to possible malicious actions preformed by other apps running on the same device (as described in [DDSW10]).

A	$\mathrm{AD}_e(A)$
1) ▦	☠
2) $\delta \gg_\gamma$ 🔐$^{(?)}$[O] $\gg_\gamma \delta'$	☠
3) $\delta \gg_\gamma$ 🔐$^{(?)}$[$\{O\}\cup$F] $\gg_\gamma \delta'$	$\not\varnothing$

Table 4.2: Capabilites of AD.

Moreover, private keys and sensitive data are usually kept in specific software wallets and keystores, but many flaws on such software objects have been found and exploited during the last years (see [ST16, GPP+16]).

The effects of AD on the authenticators are reported in Table 4.2.

It is worth noting that, in this work, we do not consider the possibility that an attacker can clone a SIM card. To model such capability, it would be possible to add $AD(\delta \gg_\gamma$ 🖬$^{(?)} \gg_{\gamma'} \delta') =$ ☠ to Table 4.2.

Lines 1 and 2 of Table 4.2. AD can neutralize Look-up Secrets (by taking a picture of them) and software authenticators only attesting ownership factors (e.g., 🔐[O]). Indeed, being able to

interact with the authenticator in place of the user, AD can provide the required input to the authenticator (as an `opid`) and obtain the authenticator output for sending it to the server.

Lines 3 of Table 4.2. AD can partially compromise ownership factor of multi-factor software authenticators, e.g., $AD_e(\delta \gg_\gamma$ [O,K] $\gg_\gamma \delta') = \delta \gg_\gamma$ [K] $\gg_\gamma \delta'$. This because it can substitute the User in the communications with A_i, hence getting access to it and having the possibility to prove its possession (i.e., the Ownership factor). However, as for DT, we assume that AD cannot compromise knowledge and inherence factors.[3] Therefore, it will not e able to produce the other factors (Knowledge or Inherence ones) that are necessary for unlocking the authenticator functionality. Notice that the same rules apply to out-of-band software authenticators, i.e., .

It is worth noting that we assume authenticator devices to be duplication-proof by construction (as most devices include some secure hardware element). Therefore, they cannot be attacked by AD.

4.1.1.3 Shoulder Surfer (SS)

A *Shoulder Surfer* is an attacker that peers over a victim's shoulder to see what he or she is up to (as presented in [LM11]). Such an attacker model is widely recognized as a relevant threat, given the increasing usage of mobile devices (where the user inputs sequences and passwords in plain sight) and the possibility to perform the malicious actions (i.e., peeking) without technical knowledge. Many studies (as [MMvZEF17, LB19, ALFD16, JFR17]) tried to evaluate the risks and possible mitigation of this attacker model, but solutions are not always feasible to perform (as for example [AMvZE+14]). In [NIS17], SS is defined as a sub-case of the eavesdropper attacker.

In our analysis, we consider SS able to target any input or datum that appears on his sight (as presented in [LM11]). This means that he can successfully attack authenticators relying on a knowledge factor and authenticator outputs that are manually inserted on the endpoint.

The effects of SS on the authenticators are reported in Table 4.3.

A	$SS_e(A)$
1)	☠
2)	☠
3) [F] $\gg_h$ `otp`	☠
4) $\delta \gg_\gamma$ $^{(?)}$[K] $\gg_\gamma \delta'$	☠
5) $\delta \gg_\gamma$ $^{(?)}$[{K}∪F] $\gg_\gamma \delta'$	~~☠~~
6) [F] $\gg_h$ `otp`	☠
7) $\delta \gg_\gamma$ $^{(?)}$[K] $\gg_\gamma \delta'$	☠
8) $\delta \gg_\gamma$ $^{(?)}$[{K}∪F] $\gg_\gamma \delta'$	~~☠~~

Table 4.3: Capabilities of SS.

Lines 1 and 2 of Table 4.3. SS neutralizes and , since he can overlook the user when manually inserting the secrets on the endpoint.

Lines 3 and 6 of Table 4.3. SS can successfully attack authenticators that *(i)* generate an `otp` which is not specifically bound to an operation (i.e., it has been not generated from an `opid`) and *(ii)*

[3]Note that the NIST [NIS17] assumes that AD can also clone a memorized secret since the user might have annotated it on paper. Here we neglect this case as it would reduce a memorized secret to a look-up secret.

require the user to manually copy the otp on the Endpoint.

In these specific cases, indeed, the attacker can compromise the otp that the user inserts on the endpoint and reuse it for authorizing his malicious operation, therefore compromising the whole authenticator which generated that otp. It is worth noting that the same attack will not be effective on those otps that have been generated basing on an opid: since this kind of otp is bound to a specific operation (requested by user), even if the attacker compromises it, SS would not be able to use it for his malicious session.

Lines 4 and 7 of Table 4.3. SS can compromise single-factor, knowledge based authenticators, e.g., $\delta \gg_\gamma$ [K] $\gg_\gamma \delta'$ and $\delta \gg_\gamma$ [K] $\gg_\gamma \delta'$, and partially compromise multi-factor authenticators relying on a knowledge factor (by removing it - ~~K~~). It is worth noting that this also includes out-of-band software, i.e., , since they are a specific sub-case. For example, SS(opid$\gg_n$ [O,K] $\gg_h$ otp) = opid$\gg_n$ [O] $\gg_h$ otp.

Lines 5 and 8 of Table 4.3. SS can oversee the input of the secret code/PIN on the keypad of an authenticator. Therefore, the Knowledge factor of such authenticators is compromised.

4.1.1.4 Eavesdropping Software (ES)

An *Eavesdropping Software* is a program that runs on the user endpoint and communicates with the attacker. In particular, such a software aims at collecting any data passing through the endpoint, as a keystroke on a keyboard or the content of an SMS received by a mobile device.

This kind of threat has been widely used during the last years and features a vast set of different software to be employed to execute the attack. The most common examples of such softwares are spywares hidden in browser extensions (as shown in [AVZ+18]) and keyloggers (as HawkEye and Olympic Vision, described in [EU19] and [Mae16]) - especially targeting mobile devices (as presented in [MS13, CCK15]).

A	ES$_e(A)$
1)	
2)	
3) [F] $\gg_h$ otp	
4) [F] $\gg_m$ otp	†
5) otp$\gg_\gamma$ [F] $\gg_m$ otp	†
6) otp$\gg_m$ [F] $\gg_\gamma$ otp	†
7) [F] $\gg_h$ otp	
8) $\delta \gg_\gamma$ [K] $\gg_\gamma \delta'$	*
9) $\delta \gg_\gamma$ [{K}∪F] $\gg_\gamma \delta'$	~~K~~*

† when $e = App$
* when $(e = Browser \wedge RunsOn(X) = Desktop)$ or $(e = App \wedge RunsOn(X) = Mobile)$

Table 4.4: Capabilities of ES.

Therefore, in our modeling, we assume ES can read, but not modify, data exchanged between the user and her endpoint. Moreover, ES can eavesdrop data transmitted via the telephony network when installed on a smartphone working as Endpoint (since a software with enough privileges can interact with the telephony network APIs, as shown in [DMC15]).

The effects of ES on the authenticators are presented in Table 4.4.

Lines 1 and 2 of Table 4.4. As for SS, ES neutralizes 🔑 and ▦, since they are manually input on the endpoint through the keyboard, which is observed by the attacker.

Lines 3, 4, 5, 6 and 7 of Table 4.4. As for SS, ES can neutralize authenticators providing the user with an `otp` which has been generated indipendently form an `opid` and that must be manually inserted into the endpoint. Additionally, ES can eavesdrop data transmitted over the telephony network, when $e = App$.

Lines 8 and 9 of Table 4.4. ES can eavesdrop the knowledge factors (e.g., PIN, secret code) which are manually inserted for unlocking software authenticators running on the endpoint.

4.1.1.5 Social Engineer (SE)

A *Social Engineer* is an attacker that "manipulates a person to take an action that may or may not be in the target's best interest" [Had10]. To this aim, he may adopt the most diverse techniques, ranging from various kinds of phishing attacks (as the ones described in [CCR16, HMN15]) to more "physical" and social approaches (see [KHHW15]).

Social Engeneering attacks, which started more than 20 years ago, are still of notable relevance. According to [Kas19], the average share of spam in global mail traffic, in the first part of 2019, amounted to 57.64%, while 130 million redirects - derived from phishing attempts - have been detected. Moreover, in 2017, a report deployed by Verizon ([Ver17]) stated that 90% of incidences and breaches included a phishing element.

A	$\text{SE}_e(A)$
1) 🔑	☠
2) ▦	☠
3) $\delta \gg_\gamma$ 🖩[F] $\gg_\gamma \delta'$	☠
4) $\delta \gg_\gamma$ 🗣[F] $\gg_\gamma \delta'$	☠

Table 4.5: Capabilities of SE.

In our analysis, we assume that the attacker can induce the user to perform unwanted actions leveraging all the possible methods that have previously mentioned. For instance, we assume that the user can be fooled to reveal confidential data to the attacker by inserting them on a phishing website. Furthermore, we assume that SE can induce the user to generate and communicate him an `otp`.

The effects of SE on the authenticators are presented in Table 4.5.

Lines 1 and 2 of Table 4.5. SE neutralizes 🔑 and ▦, since it can collect the secrets by inducing the user to insert them in a malicious web page.

Lines 3 and 4 of Table 4.5. SE can successfully attack any hardware/software authenticator not showing information regarding the operation that the user is going to authorize. Indeed, SE can induce the user to produce any `otp`, by providing her the proper input. Since the user cannot be aware of being generating an `otp` that can be used for another (malicious) operation, the attack is successful.

However, we assume that SE becomes ineffective when the user is aware of the ongoing opera-

tion, i.e., when an authenticator is labeled with **?**. This because, as mentioned in Section 3.2.4, we assume that the information displayed on the authenticator is enough to allow the user to distinguish one operation (e.g., her operation) from another (e.g., one associated to the attacker).

4.1.1.6 Man-in-the-Browser (MB)

The *Man in the Browser* is a threat that compromises the browser of the user, usually by means of trojan horses. As reported in [Güh06], MB modifies communications on-the-fly (as they are formed in browsers), manipulating what the user is shown and modifying the responses to the server. This kind of software is usually disguised as Browser Helper Objects, browser extensions or it is injected as javascript code through ajax calls.

This kind of attack is often considered to be a specific case of the Man-in-the-Middle (MitM) attack, acting between the user and her browser. However, as described in [DC12], MB differs from MitM in many ways, especially in terms of capabilities: while the MitM is only guaranteed to be able to handle public key encryption, MitB is "immune" to all forms of channel encryption, by being external to it. Indeed, by infecting the browser, it can handle messages after the have been decoded by the browser.

It is worth noting that, with respect to what defined in [NIS17], here we distinguish between MB and MM (see below) as a refinement of the generic *endpoint compromiser*. Following this distinction, we assume that MB can only act on browsers installed on desktop computers. In our analysis, we consider that any piece of data displayed and typed on the browser can be intercepted and modified by MB.

A	$\text{MB}_e(A)$
1) 🔑	☠*
2) ▦	☠*
3) $\delta \gg$ 🖩[F] $\gg_\gamma \delta'$	☠*
4) $\delta \gg$ 🔐[F] $\gg_\gamma \delta'$	☠*

*Only applies when $e = \textit{Browser}$

Table 4.6: Capabilities of MB.

The effects of MB on the authenticators are reported in Table 4.6.

MB neutralizes 🔑, ▦, $\delta \gg_\gamma$ 🖩[F] $\gg_\gamma \delta'$ and $\delta \gg_\gamma$ 🔐[F] $\gg_\gamma \delta'$ when the Endpoint from which the protocol is started is a browser. This because it can interrupt, eavesdrop and modify any message which is sent or received by the Endpoint. For instance, MB can eavesdrop the memorized secret and reuse it for a parallel, malicious session. Otherwise, it can induce the user to generate or leak an `otp` that could be used for its malicious actions.

As mentioned in [TFT+16, Ayy17], possible countermeasures to such a powerful threat model include the adoption of dedicated hardware devices or specifically hardened software authenticators, reporting clear information regarding the operation that the user is about to authorize. For this reason, in our modeling, MB does not affect authenticators that show the ongoing operation (**?**), since they allow the user to revise the authenticator input and block the execution of the MFA protocol when necessary.

4.1.1.7 Man-in-the-Mobile (MM)

A *Man in the Mobile* controls the mobile device of the user. This threat model is the counterpart - for mobile devices - of MB. Therefore, as for MB, MM leverages a malware (or a combination of them, colluding for targeting different aspects of the mobile platform - as shown in [MA15]) to gain full control of inputs and outputs. In our modeling, we assume MM can leverage specific malwares (as Riltok [Shi19]), privilege escalation techniques (as presented in [XPW+14]) and anything that allows for getting the possibility to read and modify the user inputs, the displayed information and the responses to the server.

The effects of MM on the authenticators are reported in Table 4.7.

Lines 1-3 of Table 4.7. When an app on a mobile device is the endpoint of the MFA protocol, MM neutralizes the same authenticators as MB, i.e., 🔑, ▦, $\delta \gg_\gamma$ 🖩[F] $\gg_\gamma \delta'$ and $\delta \gg_\gamma$ 🔐[F] $\gg_\gamma \delta'$.

Line 4 of Table 4.7. MM neutralizes software authenticators that run on a compromised device even when the endpoint is a Browser computer. Thus, MM neutralizes both $\delta \gg_\gamma$ 🔐[F] $\gg_\gamma \delta'$ and $\delta \gg_\gamma$ 🔐?[F] $\gg_\gamma \delta'$ if they run on a mobile device, e.g., a smartphone.

Finally, as for MB, the solutions for mitigating the attack include the usage of external hardware devices informing the user on the ongoing operations. Therefore, MM is not effective against $\delta \gg_\gamma$ 🖩?[F] $\gg_\gamma \delta'$.

A	$\mathrm{MM}_e(A)$
1) 🔑	☠*
2) ▦	☠*
3) $\delta \gg_\gamma$ 🖩[F] $\gg_\gamma \delta'$	☠*
4) $\delta \gg_\gamma$ 🔐(?)[F] $\gg_\gamma \delta'$	☠†

*Only applies when $e = App$.
† Only when $RunsOn(A) = Mobile$

Table 4.7: Capabilities of MM.

4.1.1.8 Combination of attackers

In our analysis, we consider both single attacker models and their combination. Specifically, we combine the attacker models presented above by aggregating the respective capabilities. We indicate such combination with symbol $\circ$, e.g., $\mathrm{AD}_e \circ \mathrm{SS}_e$ denotes the combination of Authenticator Duplicator and Shoulder Surfer. To exemplify such attacker models, consider Nordea1, that can be written as $P =$ 🔑;`opid`$\gg_h$ 🖩[O,K] $\gg_h$ `otp`. It is neutralized by $\mathrm{SS}_{Browser} \circ \mathrm{DT}_{Browser}$. As a matter of fact, $\mathrm{SS}_{Browser} \circ \mathrm{DT}_{Browser}(P) = DT_{Browser}(SS_{Browser}(P)) = \mathrm{DT}_{Browser}($☠;`opid`$\gg_h$ 🖩[O] $\gg_h$ `otp`$) =$ ☠;☠ = ☠.

4.1.1.9 Compliance w.r.t. NIST attackers

In [NIS17] a list of attacker models is provided. Here we put their attackers in correspondence with the attacker models presented in the previous sections. Table 4.8 reports the list of the

Table 4.8: Attacker models defined by NIST covered by our framework.

Assertion Manufacture or Modification	Theft	Duplication	Eaves-dropping	Offline Cracking	Side Channel Attack	Phishing or Pharming	Social Engineering	Online Guessing	Endpoint Compromise	Unauthorized Binding
✖	✔	✔	✔	✖	✖	✔	✔	✖	✔	✔*

attackers of [NIS17]. We denote by ✔ those that are currently supported by our framework.

Four attackers are not supported (✖). This is due to our working assumptions. In particular, our approach is designed to only rely on the user experience. Thus, the expected input only includes details about the MFA protocol that a (non expert) user can observe. This entails, for instance, that we cannot model the internal structure of an authenticator device or software.

The assertion manufacture and modification attack applies to the service infrastructure that handles the authentication process. Such infrastructure is usually totally transparent to the user. Also, attackers such as offline cracking, side channel and online guessing exploit flaws in the implementation of the authenticators. Under our assumptions, the user is not aware of these internal and remote implementation details.

Finally, for what concerns the unauthorized binding attack, it applies to the delivery and activation of a new authenticator. This phase takes place before any execution of the MFA protocol, so it cannot be evaluated, as the other attacker models presented in this section, on the authenticators. However, a best practice that focuses on the same aspects (BP5) is defined in Section 4.2.2. Evaluating the compliance with **BP5** would give indications on the resistance against this threat model.

4.1.1.10 Considerations on user mistakes

The attackers presented in the previous sections have been modeled as functions operating various transformations on SLaMP specifications. The same approach could be used for modeling other security relevant behaviors, such as human errors.

For instance, imagine that we want to model a user not verifying the information (on the operation) she is displayed on the mobile phone, confirming all the requests. This could be modeled as a new operator, Err_{App}, that removes the notation $^{?}$ (when present) from the authenticators running on her phone (as shown in Table 4.9). Although being of relevance, this kind of transformations are out of the scope of this work and, thus, account as future work.

A	$Err_{App}(A)$
$\delta \gg_\gamma \tau^{?}[\mathrm{F}] \gg_\gamma \delta'$	$\delta \gg_\gamma \tau[\mathrm{F}] \gg_\gamma \delta'^{\dagger}$
† when $RunsOn(X) = Mobile$	

Table 4.9: Capabilites of Err_{App}.

4.1.2 Attackers on SPS

In this section, we present how the attackers presented in Section 4.1.1 intervene on an SPS specification.

In our framework, attackers are modeled as a modification of the security properties of the communication channels involved in the protocol execution. This means that an attacker affects the SPS actions of the targeted authenticator by modifying the security properties of one or more communication channels.

As for the attackers of SLaMP, we define $f^S : \{Browser, App\} \times \text{SPS} \rightarrow \text{SPS}$. If $A_k =$  , its corresponding SPS actions would allow an intruder to obtain the needed data pieces and bypass its protection, keeping executing the protocol.

In the following sections, we put forward the attacker definitions. In Tables 4.10, 4.11 and 4.12, the effects of the attackers on SPS are reported.

4.1.2.1 Device Thief

In our analysis, we model DT_e^S by altering some security properties of the channels involving the Authenticator, the User and the Endpoint. Firstly, both the channel between the User and the Authenticator and the one between the Endpoint and the Authenticator lose the property of authenticity. Secondly, the channels between the Authenticator and the User and between the Authenticator and the Endpoint lose the confidentiality property. In terms of SPS specification, this means that $\texttt{U} \bullet\!\!\rightarrow\!\!\bullet\ \texttt{A}_k$, $\texttt{E} \bullet\!\!\rightarrow\!\!\bullet\ \texttt{A}_k$, become $\texttt{U} \rightarrow\!\!\bullet\ \texttt{A}_k$ and $\texttt{E} \rightarrow\!\!\bullet\ \texttt{A}_k$. Moreover, $\texttt{A}_k \bullet\!\!\rightarrow\!\!\bullet\ \texttt{U}$ and $\texttt{A}_k \bullet\!\!\rightarrow\!\!\bullet\ \texttt{E}$ become $\texttt{A}_k \bullet\!\!\rightarrow\ \texttt{U}$ and $\texttt{A}_k \bullet\!\!\rightarrow\ \texttt{E}$.

4.1.2.2 Authenticator Duplicator

In our modeling, the copy of an authenticator maps to the fact that a malicious attacker can interact with it: as for DT, we model AD_e^S by (i) dropping the property of authenticity of **U to A** and **E to A** and (ii) dropping the property of confidentiality of **A to U** and **A to E**. In terms of SPS specification, this means that $\texttt{U} \bullet\!\!\rightarrow\!\!\bullet\ \texttt{A}_k$, $\texttt{E} \bullet\!\!\rightarrow\!\!\bullet\ \texttt{A}_k$, become $\texttt{U} \rightarrow\!\!\bullet\ \texttt{A}_k$ and $\texttt{E} \rightarrow\!\!\bullet\ \texttt{A}_k$. Moreover, $\texttt{A}_k \bullet\!\!\rightarrow\!\!\bullet\ \texttt{U}$ and $\texttt{A}_k \bullet\!\!\rightarrow\!\!\bullet\ \texttt{E}$ become $\texttt{A}_k \bullet\!\!\rightarrow\ \texttt{U}$ and $\texttt{A}_k \bullet\!\!\rightarrow\ \texttt{E}$. It is worth noting that the described transformations apply only if $Platform(Ak) \neq Hardware$ (since AD cannot replicate Hardware authenticators).

Table 4.10: Effects of attackers DT_e^S, AD_e^S, SS_e^S, ES_e^S.

f_e^S	σ	$f_e^S(\sigma)$
DT_e^S	$\ldots$; U $\bullet\rightarrow\bullet$ $\mathrm{A_k}$: m; $\ldots$	$\ldots$; U $\rightarrow\bullet$ $\mathrm{A_k}$: m; $\ldots$
	$\ldots$; $\mathrm{A_k}$ $\bullet\rightarrow\bullet$ U: m; $\ldots$	$\ldots$; $\mathrm{A_k}$ $\bullet\rightarrow$ U: m; $\ldots$
	$\ldots$; E $\bullet\rightarrow\bullet$ $\mathrm{A_k}$: m; $\ldots$	$\ldots$; E $\rightarrow\bullet$ $\mathrm{A_k}$: m; $\ldots$
	$\ldots$; $\mathrm{A_k}$ $\bullet\rightarrow\bullet$ E: m; $\ldots$	$\ldots$; $\mathrm{A_k}$ $\bullet\rightarrow$ E: m; $\ldots$
AD_e^S	$\ldots$; U $\bullet\rightarrow\bullet$ $\mathrm{A_k}$: m; $\ldots$	$\ldots$; U $\rightarrow\bullet$ $\mathrm{A_k}$: m; $\ldots$ †
	$\ldots$; $\mathrm{A_k}$ $\bullet\rightarrow\bullet$ U: m; $\ldots$	$\ldots$; $\mathrm{A_k}$ $\bullet\rightarrow$ U: m; $\ldots$ †
	$\ldots$; E $\bullet\rightarrow\bullet$ $\mathrm{A_k}$: m; $\ldots$	$\ldots$; E $\rightarrow\bullet$ $\mathrm{A_k}$: m; $\ldots$ †
	$\ldots$; $\mathrm{A_k}$ $\bullet\rightarrow\bullet$ E: m; $\ldots$	$\ldots$; $\mathrm{A_k}$ $\bullet\rightarrow$ E: m; $\ldots$ †
SS_e^S	$\ldots$; U $\rightarrow\bullet$ E: m; $\ldots$	$\ldots$; U $\rightarrow$ E: m; $\ldots$
	$\ldots$; E $\bullet\rightarrow\bullet$ U: m; $\ldots$	$\ldots$; E $\bullet\rightarrow$ U: m; $\ldots$
	$\ldots$; U $\bullet\rightarrow\bullet$ $\mathrm{A_k}$: m; $\ldots$	$\ldots$; U $\bullet\rightarrow$ $\mathrm{A_k}$: m; $\ldots$ ‡
	$\ldots$; $\mathrm{A_k}$ $\bullet\rightarrow\bullet$ U: m; $\ldots$	$\ldots$; $\mathrm{A_k}$ $\bullet\rightarrow$ U: m; $\ldots$
$\mathrm{ES}_{Browser}^S$	$\ldots$; U $\rightarrow\bullet$ E: m; $\ldots$	$\ldots$; U $\rightarrow$ E: m; $\ldots$
	$\ldots$; E $\bullet\rightarrow\bullet$ U: m; $\ldots$	$\ldots$; E $\bullet\rightarrow$ U: m; $\ldots$
	$\ldots$; U $\bullet\rightarrow\bullet$ $\mathrm{A_k}$: m; $\ldots$	$\ldots$; U $\bullet\rightarrow$ $\mathrm{A_k}$: m; $\ldots$ *,‡
	$\ldots$; $\mathrm{A_k}$ $\bullet\rightarrow\bullet$ U: m; $\ldots$	$\ldots$; $\mathrm{A_k}$ $\bullet\rightarrow$ U: m; $\ldots$ $*$
ES_{App}^S	$\ldots$; U $\rightarrow\bullet$ E: m; $\ldots$	$\ldots$; U $\rightarrow$ E: m; $\ldots$
	$\ldots$; E $\bullet\rightarrow\bullet$ U: m; $\ldots$	$\ldots$; E $\bullet\rightarrow$ U: m; $\ldots$
	$\ldots$; U $\bullet\rightarrow\bullet$ $\mathrm{A_k}$: m; $\ldots$	$\ldots$; U $\bullet\rightarrow$ $\mathrm{A_k}$: m; $\ldots$ $\diamond$
	$\ldots$; $\mathrm{A_k}$ $\bullet\rightarrow\bullet$ U: m; $\ldots$	$\ldots$; $\mathrm{A_k}$ $\bullet\rightarrow$ U: m; $\ldots$ $\diamond$
	$\ldots$; T $\bullet\rightarrow\bullet$ $\mathrm{A_k}$: m; $\ldots$	$\ldots$; T $\bullet\rightarrow$ $\mathrm{A_k}$: m; $\ldots$
	$\ldots$; $\mathrm{A_k}$ $\bullet\rightarrow\bullet$ T: m; $\ldots$	$\ldots$; $\mathrm{A_k}$ $\bullet\rightarrow$ T: m; $\ldots$
† only if $Platform(Ak) \neq Hardware$, ‡ only if $m \neq$ fI(U,Ak)		
$*$ only if $Platform(Ak) = Desktop$, $\diamond$ only if $Platform(Ak) = App$		

4.1.2.3 Shoulder Surfer

SS compromises any datum on its sight. Therefore we model SS_e^S by removing the confidentiality property of *a)* the channels between the User and the Endpoint and *b)* the channels between the User and the Authenticator. In terms of SPS specification, this means that the actions including U $\rightarrow\bullet$ E, E $\bullet\rightarrow\bullet$ U, U $\bullet\rightarrow\bullet$ A*k* and A*k* $\bullet\rightarrow\bullet$ U would be with as U $\rightarrow$ E, E $\bullet\rightarrow$ U, U $\bullet\rightarrow$ A*k* and A*k* $\bullet\rightarrow$ U, respectively.

Table 4.11: Effects of attackers SE_e^S and MB_e^S.

f_e^S	σ	$f_e^S(\sigma)$
SE_e^S	...; S $\bullet\rightarrow\bullet$ E: m; ...	...; S $\rightarrow\bullet$ E: m; ...
	...; E $\rightarrow\bullet$ S: m; ...	...; E $\rightarrow$ S: m; ...
	...; S $\bullet\rightarrow\bullet$ A_k: m; ...	...; S $\rightarrow\bullet$ A_k: m; ...
	...; A_k $\bullet\rightarrow\bullet$ S: m; ...	...; A_k $\bullet\rightarrow$ S: m; ...
	...; T $\bullet\rightarrow\bullet$ A_k: m; ...	...; T $\rightarrow\bullet$ A_k: m; ...
	...; A_k $\bullet\rightarrow\bullet$ T: m; ...	...; A_k $\bullet\rightarrow$ T: m; ...
$MB_{Browser}^S$	...; U $\rightarrow\bullet$ E: m; ...	...; U $\rightarrow$ E: m; ...
	...; E $\bullet\rightarrow\bullet$ U: m; ...	...; E $\rightarrow$ U: m; ...
	...; E $\rightarrow\bullet$ S: m; ...	...; E $\rightarrow$ S: m; ...
	...; S $\bullet\rightarrow\bullet$ E: m; ...	...; S $\rightarrow$ E: m; ...
	...; E $\bullet\rightarrow\bullet$ A_k: m; ...	...; E $\rightarrow$ A_k: m; ...
	...; A_k $\bullet\rightarrow\bullet$ E: m; ...	...; A_k $\rightarrow$ E: m; ...

4.1.2.4 Eavesdropping Software

In our analysis, we model ES_e^S by removing the confidentiality property of *a)* the channel between the User and the Endpoint, *b)* the channel between the User and the Software Authenticator installed on the endpoint and *c)* the channel between the Telephony Server and the Endpoint (only in the case that the Endpoint is a smartphone). In terms of SPS specification, this means that the actions including U $\rightarrow\bullet$ E, E $\bullet\rightarrow\bullet$ U, U $\bullet\rightarrow\bullet$ A_k and A_k $\bullet\rightarrow\bullet$ U would include U $\rightarrow$ E, E $\bullet\rightarrow$ U, U $\bullet\rightarrow$ A_k and A_k $\bullet\rightarrow$ U instead. Note that the modifications on the actions involving *Ak* only if either $e = Browser \wedge Platform(A_k) = Desktop$ or $e = App \wedge Platform(A_k) = Mobile$.

4.1.2.5 Social Engineer

In our analysis, SE_e^S is modeled by dropping the confidentiality property of the communication channel between the Endpoint and the Server and dropping the authentication property of the channel between the Server and the Endpoint. In SPS, this means that E $\bullet\rightarrow\bullet$ S becomes E $\bullet\rightarrow$ S and S $\bullet\rightarrow\bullet$ E becomes S $\rightarrow\bullet$ E. Moreover, the same applies for the channels between the servers (S and T) and the $k-th$ authenticator.

4.1.2.6 Man-in-the-Browser

In our analysis, we consider that any piece of data displayed and typed on the browser can be intercepted and modified by MB. The user is potentially displayed incorrect information.

Table 4.12: Effects of MM_e^S.

f_e^S	σ	$f_e^S(\sigma)$
$\text{MM}^S_{Browser}$	$\ldots$; T $\bullet \rightarrow \bullet$ A_k: m; $\ldots$	$\ldots$; T $\rightarrow$ A_k: m; $\ldots$
	$\ldots$; A_k $\bullet \rightarrow \bullet$ T: m; $\ldots$	$\ldots$; A_k $\rightarrow \bullet$ T: m; $\ldots$
	$\ldots$; E $\bullet \rightarrow \bullet$ A_k: m; $\ldots$	$\ldots$; E $\rightarrow$ A_k: m; $\ldots *$
	$\ldots$; A_k $\bullet \rightarrow \bullet$ E: m; $\ldots$	$\ldots$; A_k $\rightarrow$ E: m; $\ldots *$
	$\ldots$; S $\bullet \rightarrow \bullet$ A_k: m; $\ldots$	$\ldots$; S $\rightarrow$ A_k: m; $\ldots *$
	$\ldots$; A_k $\bullet \rightarrow \bullet$ S: m; $\ldots$	$\ldots$; A_k $\rightarrow \bullet$ S: m; $\ldots *$
	$\ldots$; U $\bullet \rightarrow \bullet$ A_k: m; $\ldots$	$\ldots$; U $\rightarrow$ A_k: m; $\ldots *$
	$\ldots$; A_k $\bullet \rightarrow \bullet$ U: m; $\ldots$	$\ldots$; A_k $\rightarrow$ U: m; $\ldots *$
MM^S_{App}	$\ldots$; U $\rightarrow \bullet$ E: m; $\ldots$	$\ldots$; U $\rightarrow$ E: m; $\ldots$
	$\ldots$; E $\bullet \rightarrow \bullet$ U: m; $\ldots$	$\ldots$; E $\rightarrow$ U: m; $\ldots$
	$\ldots$; E $\rightarrow \bullet$ S: m; $\ldots$	$\ldots$; E $\rightarrow$ S: m; $\ldots$
	$\ldots$; S $\bullet \rightarrow \bullet$ E: m; $\ldots$	$\ldots$; S $\rightarrow$ E: m; $\ldots$
	$\ldots$; E $\bullet \rightarrow \bullet$ A_k: m; $\ldots$	$\ldots$; E $\rightarrow$ A_k: m; $\ldots$
	$\ldots$; A_k $\bullet \rightarrow \bullet$ E: m; $\ldots$	$\ldots$; A_k $\rightarrow$ E: m; $\ldots$
	$\ldots$; T $\bullet \rightarrow \bullet$ A_k: m; $\ldots$	$\ldots$; T $\rightarrow$ A_k: m; $\ldots$
	$\ldots$; A_k $\bullet \rightarrow \bullet$ T: m; $\ldots$	$\ldots$; A_k $\rightarrow \bullet$ T: m; $\ldots$
	$\ldots$; S $\bullet \rightarrow \bullet$ A_k: m; $\ldots$	$\ldots$; S $\rightarrow$ A_k: m; $\ldots *$
	$\ldots$; A_k $\bullet \rightarrow \bullet$ S: m; $\ldots$	$\ldots$; A_k $\rightarrow \bullet$ S: m; $\ldots *$
	$\ldots$; U $\bullet \rightarrow \bullet$ A_k: m; $\ldots$	$\ldots$; U $\rightarrow$ A_k: m; $\ldots *$
	$\ldots$; A_k $\bullet \rightarrow \bullet$ U: m; $\ldots$	$\ldots$; A_k $\rightarrow$ U: m; $\ldots *$
$*$ only if $Platform(Ak) = Mobile$		

In terms of SPS, this means that E would be communicating with other roles leveraging the following channels: $\texttt{E} \rightarrow \texttt{S}$, $\texttt{S} \rightarrow \texttt{E}$, $\texttt{U} \rightarrow \texttt{E}$, $\texttt{E} \rightarrow \texttt{U}$, $\texttt{E} \rightarrow \texttt{A}_k$ and $\texttt{A}_k \rightarrow \texttt{E}$.

4.1.2.7 Man-in-the-Mobile

The Man-in-the-Mobile is the counterpart of MB for the App endpoint. In terms of SPS, this means that the mobile endpoint E would be communicating with other roles leveraging the following channels: $\texttt{E} \rightarrow \texttt{S}$, $\texttt{S} \rightarrow \texttt{E}$, $\texttt{U} \rightarrow \texttt{E}$, $\texttt{E} \rightarrow \texttt{U}$, $\texttt{E} \rightarrow \texttt{A}_k$ and $\texttt{A}_k \rightarrow \texttt{E}$. However, even when the endpoint is a Desktop computer, MM can still compromise authenticators installed on the smartphone. In this case, we model this behavior by dropping all the security properties of the channels between such authenticators and other roles.

4.1.3 Partial Correctness

In this section we provide a proof of formal correctness of our attackers. In particular, we aim to prove the following.

Theorem 1. *Given P, f_e s.t. $f_e(P) =$ 🏴‍☠️ then $\not\vdash f_e^S(Sem(P))$*

Proof. By modus ponens, we assume the premise and we prove the conclusion. Moreover, we notice that for each protocol $P = A_1; \ldots; A_n$, holds that $f_e(P) =$ 🏴‍☠️ if and only if $\forall i \in \{1, ..., n\}. f_e(A_i) =$ 🏴‍☠️. Below, we prove the partial correctness w.r.t. each authenticator, namely if $f_e(A) =$ 🏴‍☠️ then $\not\vdash f_e^S(Sem(A))$, $\forall A \in \Omega$.

Memorized Secrets
In this case, $f_e($🔑$) =$ 🏴‍☠️. According to the tables in Section 4.1.1, this happens for $f \in \{$SS, ES, SE, MB, MM$\}$. Moreover, $\sigma = Sem($🔑$) = \ldots; PRE \cdot \mathtt{U} \to \bullet\mathtt{E} : \mathtt{mem}(\mathtt{U},\mathtt{S},\mathtt{k}) \cdot \mathtt{E} \to \bullet\mathtt{S} : \mathtt{mem}(\mathtt{U},\mathtt{S},\mathtt{k}); GOAL$, where *PRE* and *GOAL* are defined as in Section 3.3.

Thus, we have five sub-cases.

SS By definition of SS_e^S in Table 4.10, $\mathtt{U} \to \bullet\mathtt{E} : \mathtt{mem}(\mathtt{U},\mathtt{S},\mathtt{k})$ becomes $\mathtt{U} \to \mathtt{E} : \mathtt{mem}(\mathtt{U},\mathtt{S},\mathtt{k})$.
Therefore we compute: $\mathrm{SS}_e^S(\sigma) = \ldots; PRE \cdot \mathtt{U} \to \mathtt{E} : \mathtt{mem}(\mathtt{U},\mathtt{S},\mathtt{k}) \cdot \mathtt{E} \to \bullet\mathtt{S} : \mathtt{mem}(\mathtt{U},\mathtt{S},\mathtt{k}); GOAL$.
Since $\mathtt{mem}(\mathtt{U},\mathtt{S},\mathtt{k})$ is accessible to the attacker, $\not\vdash SS_e^S(\sigma)$.

ES This case is symmetric to the previous one.

SE By definition of SE_e^S in Table 4.10, $\mathtt{E} \to \bullet\mathtt{S} : \mathtt{mem}(\mathtt{U},\mathtt{S},\mathtt{k})$ becomes $\mathtt{E} \to \mathtt{S} : \mathtt{mem}(\mathtt{U},\mathtt{S},\mathtt{k})$.
Thus, we compute $\mathrm{SE}_e^S(\sigma) = \ldots; PRE \cdot \mathtt{U} \to \bullet\mathtt{E} : \mathtt{mem}(\mathtt{U},\mathtt{S},\mathtt{k}) \cdot$
$\mathtt{E} \to \bullet\mathtt{S} : \mathtt{mem}(\mathtt{U},\mathtt{S},\mathtt{k}); GOAL$. In other words, the attacker is able to collect 🔑, comprmosing it. Thus, $\not\vdash SE_e^S(\sigma)$.

MB This case applies only if $e = Browser$. In such case, by definition of $\mathrm{MB}_{Browser}^S$ in Table 4.11, $\mathtt{U} \to \bullet\mathtt{E} : \mathtt{mem}(\mathtt{U},\mathtt{S},\mathtt{k})$ becomes $\mathtt{U} \to \mathtt{E} : \mathtt{mem}(\mathtt{U},\mathtt{S},\mathtt{k})$.
Thus, we calculates $\mathrm{MB}_{Browser}^S(\sigma) = \ldots; PRE \cdot \mathtt{U} \to \mathtt{E} : \mathtt{mem}(\mathtt{U},\mathtt{S},\mathtt{k}) \cdot \ldots; GOAL$. This means that the attacker can obtain the secret, which suffices to conclude.

MM This case is symmetric to the previous one, but for $e = App$. Indeed, as shown in Table 4.7, MM does not apply to 🔑 if $e = Browser$.

Look-up Secrets

In this case, $f_e($▦$) =$ 🏴‍☠️. According to the tables in Section 4.1.1, this implies that $f \in \{$DT, AD, SS, ES, SE, MB, MM$\}$.

Moreover, $\sigma = Sem(\boxplus) = \ldots; PRE \cdot \mathtt{S} : \mathtt{NumberCoords} \cdot \mathtt{S}\bullet \rightarrow \bullet\mathtt{E} : \mathtt{Coords} \cdot \mathtt{E}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{Coords} \cdot \mathtt{U}\bullet \rightarrow \bullet\mathtt{A_k} : \mathtt{Coords} \cdot \mathtt{A_k}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords}) \cdot \mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords}) \cdot \mathtt{E} \rightarrow \bullet\mathtt{S} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords}); GOAL.$

Thus, we have seven sub-cases.

DT By definition of DT_e^S in Table 4.10, $\mathtt{U}\bullet \rightarrow \bullet\mathtt{A_k} : \mathtt{Coords}$ and $\mathtt{A_k}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}), \mathtt{Coords})$ become $\mathtt{U} \rightarrow \bullet\mathtt{A_k} : \mathtt{Coords}$ and $\mathtt{A_k}\bullet \rightarrow \mathtt{U} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords})$, respectively.
Therefore, we compute $DT_e^S(\sigma) = \ldots; PRE \cdot \mathtt{S} : \mathtt{Number} \quad \mathtt{Coords} \cdot \mathtt{S}\bullet \rightarrow \bullet\mathtt{E} : \mathtt{Coords} \cdot \mathtt{E}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{Coords} \cdot \mathtt{U} \rightarrow \bullet\mathtt{A_k} : \mathtt{Coords} \cdot \mathtt{A_k}\bullet \rightarrow \mathtt{U} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords}) \cdot \mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords}) \cdot \mathtt{E} \rightarrow \bullet\mathtt{S} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords}); GOAL.$

This suffices to hijack a session with $\mathtt{A_k}$, since an intruder can impersonate $\mathtt{U}$ in the interaction with the authenticator and obtain the needed data.

AD This case is symmetric to the previous one.

SS By definition of SS_e^S in Table 4.10, $\mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords})$ becomes $\mathtt{U} \rightarrow \mathtt{E} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords}) \cdot$
Hence, we compute $SS_e^S(\sigma) = \ldots; PRE \cdot \ldots \cdot \mathtt{U} \rightarrow \mathtt{E} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords}) \cdot \ldots;$ *GOAL*. Since $\mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords})$ is accessible to attacker, $\not\vdash SS_e^S(\sigma)$.

ES By definition of ES_e^S in Table 4.10, $\mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords})$ becomes $\mathtt{U} \rightarrow \mathtt{E} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords}) \cdot$
Hence, we compute $ES_e^S(\sigma) = \ldots; PRE \cdot \ldots \cdot \mathtt{U} \rightarrow \mathtt{E} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords}) \cdot \ldots;$ *GOAL*. Since $\mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords})$ is accessible to attacker, $\not\vdash ES_e^S(\sigma)$.

SE According to the definition of SE_e^S in Table 4.10, $\mathtt{S}\bullet \rightarrow \bullet\mathtt{E} : \mathtt{Coords}$ and $\mathtt{E} \rightarrow \bullet\mathtt{S} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords})$ become $\mathtt{S} \rightarrow \bullet\mathtt{E} : \mathtt{Coords}$ and $\mathtt{E} \rightarrow \mathtt{S} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords})$, respectively.
Therefore, we compute $SE_e^S(\sigma) = \ldots; PRE \cdot \ldots \mathtt{S} \rightarrow \bullet\mathtt{E} : \mathtt{Coords} \cdot \mathtt{E}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{Coords} \cdot \mathtt{U}\bullet \rightarrow \bullet\mathtt{A_k} : \mathtt{Coords} \cdot \mathtt{A_k}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords}) \cdot \mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords}) \cdot \mathtt{Ebullet} \rightarrow \mathtt{S} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords}); GOAL.$

This allows the attacker to collect the $\mathtt{otp}$, thus $\not\vdash SE_e^S(\sigma)$.

MB According to the definition of $MB_{Browser}^S$ in Table 4.10, $\mathtt{E}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{Coords}$ and $\mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{Coords}$ become $\mathtt{E} \rightarrow \mathtt{U} : \mathtt{Coords}$ and $\mathtt{U} \rightarrow \mathtt{E} : \mathtt{Coords}$.

Thus, we calculate $MB_{Browser}^S(\sigma) = \ldots \cdot PRE \cdot \ldots \cdot \mathtt{E} \rightarrow \mathtt{U} : \mathtt{Coords} \cdot \mathtt{U}\bullet \rightarrow \bullet\mathtt{A_k} : \mathtt{Coords} \cdot \mathtt{A_k}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords}) \cdot \mathtt{U} \rightarrow \mathtt{E} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords}) \cdot \mathtt{E} \rightarrow \mathtt{S} : \mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords}); GOAL.$

Since $\mathtt{exp}(\mathtt{lus}(\mathtt{S},\mathtt{A_k}),\mathtt{Coords})$ is accessible to attacker, $\not\vdash MB_{Browser}^S(\sigma)$.

It is worth noting that, as presented in Section 4.1.1.6, MB does not apply when $e = App$.

MM This case is symmetric to the previous one, but for $e = App$. Indeed, as shown in Table 4.7, MM applies to ▦ only if the endpoint is a mobile application.

Hardware Authenticators

In this case, $f_e(\delta \gg_\gamma \text{▦}^{(?)}[\mathrm{F}] \gg_\gamma \delta') = $ ☠.

According to the tables in Section 4.1.1, $f \in \{\text{DT, SS, ES, SE, MB, MM}\}$. Thus, we have six sub-cases.

DT in this case, we know that $F = \{O\}$. Moreover, by focusing on the definition of $Actions_k^{in}$ and $Actions_k^{out}$ given in Section 3.3.4.1, we know that $\sigma = \ldots; PRE \cdot \ldots \cdot \beta\bullet \rightarrow \bullet\mathtt{A_k} : \mathtt{m} \cdot \ldots; \mathtt{A_k}\bullet \rightarrow \bullet\beta' : \mathtt{m}' \cdot \ldots; GOAL$, where $\beta, \beta' \in \{\mathtt{U}, \mathtt{S}, \mathtt{E}, \mathtt{T}\}$, $m \in \{\mathtt{scrypt}(\mathtt{dK}(\mathtt{S}, \mathtt{A_k}), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID})), \mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k})), \mathtt{crypt}(\mathtt{pK}(\mathtt{A_k}), \mathtt{scrypt}(\mathtt{dK}(\mathtt{S}, \mathtt{A_k}), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID})))\}$ and $m' \in \{\mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k})), \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A_k})), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID})\}$. By definition of DT_e^S, $\beta\bullet \rightarrow \bullet\mathtt{A_k} : \mathtt{m}$ becomes $\beta \rightarrow \bullet\mathtt{A_k} : \mathtt{m}$ and $\mathtt{A_k}\bullet \rightarrow \bullet\beta' : \mathtt{m}'$ becomes $\mathtt{A_k}\bullet \rightarrow \beta' : \mathtt{m}'$, when $\beta, \beta' \in \{\mathtt{U}, \mathtt{E}\}$. Therefore, the attacker can successfully provide inputs to A_k and obtain the corresponding output. Thus, $\nvdash DT_e^S(\sigma)$.

SS For this attacker, we have a further distinction, according to Table 4.3.

Case ▦[F]$\gg_h$ otp In this case, by definition of $Actions_k^{out}$ given in Section 3.3.4.1, $\sigma = Sem(\text{▦}[\mathrm{F}] \gg_h \mathtt{otp}) = \ldots; PRE \cdot \ldots \cdot \mathtt{A_k}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k})) \cdot \mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k})) \cdot \ldots; GOAL$.
By definition of SS_e^S, $\mathtt{A_k}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k}))$ and $\mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k}))$ become $\mathtt{A_k}\bullet \rightarrow \mathtt{U} : \mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k}))$ and $\mathtt{U} \rightarrow \mathtt{E} : \mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k}))$.
Since $\mathtt{hash(seed(S,A}_k\mathtt{))}$ has become available to the attacker, $\nvdash SS_e^S(\sigma)$.

Case $\delta \gg_\gamma$ ▦$^{(?)}$[K]$\gg_{\gamma'} \delta'$ In this case, by definition of $Actions_k^{mid}$ given in Section 3.3.4.1, $\sigma = Sem(\delta \gg_\gamma \text{▦}^{(?)}[\mathrm{K}] \gg_{\gamma'} \delta') = \ldots; PRE \cdot \ldots \cdot \mathtt{U}\bullet \rightarrow \bullet\mathtt{A_k} : \mathtt{fK}(\mathtt{U}, \mathtt{A_k}) \cdot \ldots; GOAL$.
By definition of SS_e^S, $\mathtt{U}\bullet \rightarrow \bullet\mathtt{A_k} : \mathtt{fK}(\mathtt{U}, \mathtt{A_k})$ becomes $\mathtt{U}\bullet \rightarrow \mathtt{A_k} : \mathtt{fK}(\mathtt{U}, \mathtt{A_k})$.
Since $\mathtt{fK(U,A}_k\mathtt{)}$ has become available to the attacker, $\nvdash SS_e^S(\sigma)$.

ES Also in this case, we have a further distinction.

Case ▦[F]$\gg_h$ otp This case, for ES_e^S, is symmetric to the one of SS.

Case 🖥[F] $\gg_m$ otp By definition of $Actions_k^{out}$ given in Section 3.3.4.1, $\sigma = Sem(\delta \gg_\gamma$ 🖥[F] $\gg_m \mathtt{otp}) = \ldots; PRE \cdot \ldots \cdot \mathtt{A_k}\bullet \rightarrow \bullet\mathtt{T} : \mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k})) \cdot \ldots; GOAL$. $\mathrm{ES}_{Browser}^S$ cannot perform any attack. Instead, by definition of ES_{App}^S, $\mathtt{A_k}\bullet \rightarrow \bullet\mathtt{T} : \mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k}))$ becomes

$\mathtt{A_k}\bullet \rightarrow \mathtt{T} : \mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k}))$. The $\mathtt{otp}$ has become available to the attacker, hence $\not\vdash ES^S_{App}(\sigma)$.

Case $\mathtt{otp} \gg_\gamma$ [F] $\gg_m \mathtt{otp}$ As for the previous case, by definition of $Actions^{out}_k$ given in Section 3.3.4.1, $\sigma = Sem(\delta \gg_\gamma$ [F] $\gg_m \mathtt{otp}) = \ldots; PRE \cdot \ldots \cdot \mathtt{A_k}\bullet \rightarrow \bullet\mathtt{T} : \mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k})) \cdot \ldots; GOAL$.

By definition of ES^S_{App}, $\mathtt{A_k}\bullet \rightarrow \bullet\mathtt{T} : \mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k}))$ becomes $\mathtt{A_k}\bullet \rightarrow \mathtt{T} : \mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k}))$, i.e., the $\mathtt{otp}$ has become available to the attacker. Hence, $\not\vdash ES^S_{App}(\sigma)$.

Case $\mathtt{otp} \gg_m$ [F] $\gg_\gamma \mathtt{otp}$ By definition of $Actions^{in}_k$ given in Section 3.3.4.1, $\sigma = Sem(\mathtt{otp} \gg_m$ [F] $\gg_\gamma \mathtt{otp}) = \ldots; PRE \cdot \ldots \cdot \mathtt{T}\bullet \rightarrow \bullet\mathtt{A_k} : \mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k})) \cdot \ldots; GOAL$.

By definition of ES^S_{App}, $\mathtt{T}\bullet \rightarrow \bullet\mathtt{A_k} : \mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k}))$ becomes $\mathtt{T}\bullet \rightarrow \mathtt{A_k} : \mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k}))$. The $\mathtt{otp}$ has become available to the attacker, hence $\not\vdash ES^S_{App}(\sigma)$.

SE We differentiate other 4 sub-cases depending on the input/output of [F] as follows.

Case [F] $\gg_h \delta'$ By definition of $Actions^{out}_k$ of Section 3.3.4.1, we know that $\sigma = \ldots; PRE \cdot \ldots \cdot \mathtt{A_k}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k})) \cdot \mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k})) \cdot \mathtt{E} \rightarrow \bullet\mathtt{S} : \mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k})); GOAL$.

By definition of SE^S_e, $\mathtt{E} \rightarrow \bullet\mathtt{S} : \mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k}))$ becomes $\mathtt{E} \rightarrow \mathtt{S} : \mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k}))$. The attacker can hence collect the $\mathtt{otp}$ generated by the user, compromising the authenticator by proving its possession (through the $\mathtt{otp}$) to the server. Therefore $\not\vdash SE^S_e(\sigma)$.

Case $\delta \gg_h$ [F] $\gg_h \delta'$ where $\delta \neq \varepsilon$ By definitions of $Actions^{in}_k$ and $Actions^{out}_k$ of Section 3.3.4.1, we know that $\sigma = \ldots; PRE \cdot \ldots \mathtt{S}\bullet \rightarrow \bullet\mathtt{E} : \mathtt{m} \cdot \mathtt{E}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{m} \cdot \mathtt{U}\bullet \rightarrow \bullet\mathtt{A_k} : \mathtt{m} \cdot \ldots; \mathtt{A_k}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{m}' \cdot \mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{m}' \cdot \mathtt{E} \rightarrow \bullet\mathtt{S} : \mathtt{m}'; GOAL$, where $m \in \{\mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A_k}), \mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})), \mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k}))\}$ and $m' \in \{\mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k})), \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A_k})), \mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})\}$.

By definition of SE^S_e, $\mathtt{S}\bullet \rightarrow \bullet\mathtt{E} : \mathtt{m}$ and $\mathtt{E} \rightarrow \bullet\mathtt{S} : \mathtt{m}'$ become $\mathtt{S} \rightarrow \bullet\mathtt{E} : \mathtt{m}$ and $\mathtt{E} \rightarrow \mathtt{S} : \mathtt{m}'$, respectively. The attacker can pretend to be the Server, sending an $\mathtt{opid}$ relate to a malicious operation to E. At this point, the user (receiving data from E) is induced to generate an $\mathtt{otp}$ related to malicious $\mathtt{opid}$ (since she is not provided information regarding the operation). Once the $\mathtt{otp}$ is given to the Endpoint, the attacker collects it by acting (again) in place of the Server. Therefore $\not\vdash SE^S_e(\sigma)$.

Case $\delta \gg_h$ [F] $\gg_\gamma \delta'$ where $\delta \neq \varepsilon$ and $\gamma' \neq h$ By definitions of $Actions^{in}_k$ and $Actions^{out}_k$ of Section 3.3.4.1, we know that $\sigma = \ldots; PRE \cdot \ldots \mathtt{S}\bullet \rightarrow \bullet\mathtt{E} : \mathtt{m} \cdot \mathtt{E}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{m} \cdot \mathtt{U}\bullet \rightarrow \bullet\mathtt{A_k} : \mathtt{m} \cdot \ldots; \mathtt{A_k}\bullet \rightarrow \bullet\beta : \mathtt{m}' \cdot \ldots; GOAL$, where $m \in \{\mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A_k}), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID})), \mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k}))\}$, $m' \in \{\mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k})), \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A_k})), \mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})\}$ and $\beta \in \{\mathtt{E},\mathtt{T},\mathtt{S}\}$.

By definition of SE_e^S, $\mathtt{S}\bullet \to \bullet\mathtt{E} : \mathtt{m}$ and $\mathtt{A_k}\bullet \to \bullet\beta : \mathtt{m}'$ become $\mathtt{S} \to \bullet\mathtt{E} : \mathtt{m}$ and $\mathtt{A_k}\bullet \to \beta : \mathtt{m}'$, respectively. The attacker can pretend to be the Server, sending an `opid` relate to a malicious operation to E. At this point, the user (receiving data from E) is induced to generate an `otp` related to malicious `opid` (since she is not provided information regarding the operation). Once the `otp` is given to the Endpoint, the attacker collects it by acting (again) in place of the Server. Therefore, $\not\vdash SE_e^S(\sigma)$.

Case $\delta \gg_\gamma \blacksquare$**[F]** $\gg_h \delta'$ **where** $\delta \neq \varepsilon$ **and** $\gamma \neq h$ By definitions of $Actions_k^{in}$ and $Actions_k^{out}$ of Section 3.3.4.1, we know that $\sigma = \ldots; PRE \cdot \ldots \cdot \beta\bullet \to \bullet\mathtt{A_k} : \mathtt{m} \cdot \ldots \cdot$ $\mathtt{A_k}\bullet \to \bullet\mathtt{U} : \mathtt{m}' \cdot \mathtt{U} \to \bullet\mathtt{E} : \mathtt{m}' \cdot \mathtt{E} \to \bullet\mathtt{S} : \mathtt{m}'; GOAL$, where $\beta \in \{\mathtt{S}, \mathtt{E}, \mathtt{T}\}$ and $m \in \{\mathtt{scrypt}(\mathtt{dK}(\mathtt{S}, \mathtt{A_k}), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID})), \mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k}))\}$ and $m' \in \{\mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k})), \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A_k})), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID}))\}$.

By definition of SE_e^S, $\beta\bullet \to \bullet\mathtt{A_k} : \mathtt{m}$ and $\mathtt{E} \to \bullet\mathtt{S} : \mathtt{m}'$ become $\beta \to \bullet\mathtt{A_k} : \mathtt{m}$ and $\mathtt{E} \to \mathtt{S} : \mathtt{m}'$, respectively.

The attacker can hence replace β and collect the output. Therefore $\not\vdash SE_e^S(\sigma)$.

Case $\delta \gg_\gamma \blacksquare$**[F]** $\gg_{\gamma'} \delta'$ **where** $\delta \neq \varepsilon$ **and** $\gamma, \gamma' \neq h$ By definitions of $Actions_k^{in}$ and $Actions_k^{out}$ of Section 3.3.4.1, we know that $\sigma = \ldots; PRE \cdot \ldots \cdot \beta\bullet \to \bullet\mathtt{A_k} : \mathtt{m} \cdot \ldots;$ $\mathtt{A_k}\bullet \to \bullet\beta' : \mathtt{m}' \cdot \ldots; GOAL$, where $\beta, \beta' \in \{\mathtt{S}, \mathtt{E}, \mathtt{T}\}$ and $m \in \{\mathtt{scrypt}(\mathtt{dK}(\mathtt{S}, \mathtt{A_k}), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID})), \mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k}))\}$ and $m' \in \{\mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k})), \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A_k})), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID}))\}$.

By definition of SE_e^S, $\beta\bullet \to \bullet\mathtt{A_k} : \mathtt{m}$ and $\mathtt{A_k}\bullet \to \bullet\beta' : \mathtt{m}'$ become $\beta \to \bullet\mathtt{A_k} : \mathtt{m}$ and $\mathtt{A_k}\bullet \to \beta' : \mathtt{m}'$, respectively.

The attacker can hence replace β, β' and hijack the session with $A_\mathtt{k}$, giving it a malicious input and collecting the corresponding `otp`. Therefore $\not\vdash SE_e^S(\sigma)$.

MB Also in this case, we differentiate depending on the input/output of $\blacksquare$.

Case $\blacksquare$**[F]** $\gg_h \delta'$ By definition of $Actions_k^{out}$ of Section 3.3.4.1, we know that $\sigma = \ldots; PRE \cdot$ $\ldots \cdot \mathtt{A_k}\bullet \to \bullet\mathtt{U} : \mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k})) \cdot \mathtt{U} \to \bullet\mathtt{E} : \mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k})) \cdot$ $\mathtt{E} \to \bullet\mathtt{S} : \mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k})); GOAL$.

By definition of $\mathrm{MB}_{Browser}^S$, $\mathtt{U} \to \bullet\mathtt{E} : \mathtt{m}'$ becomes $\mathtt{U} \to \mathtt{E} : \mathtt{m}'$, respectively. The attacker can hence obtain the `otp`. Thus, $\not\vdash MB_{Browser}^S(\sigma)$.

Case $\delta \gg_h \blacksquare$**[F]** $\gg_h \delta'$ **where** $\delta \neq \varepsilon$ By definitions of $Actions_k^{in}$ and $Actions_k^{out}$ of Section 3.3.4.1, we know that $\sigma = \ldots; PRE \cdot \ldots \cdot \mathtt{E}\bullet \to \bullet\mathtt{U} : \mathtt{m} \cdot \mathtt{U}\bullet \to \bullet\mathtt{A_k} : \mathtt{m} \cdot \ldots \cdot \mathtt{A_k}\bullet \to \bullet\mathtt{U} : \mathtt{m}' \cdot$ $\mathtt{U} \to \bullet\mathtt{E} : \mathtt{m}' \cdot \mathtt{E} \to \bullet\mathtt{S} : \mathtt{m}'; GOAL$, where $m \in \{\mathtt{scrypt}(\mathtt{dK}(\mathtt{S}, \mathtt{A_k}), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID})), \mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k}))\}$ and $m' \in \{\mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k})), \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A_k})), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID}))\}$.

By definition of MB_e^S, $\mathtt{E}\bullet \to \bullet\mathtt{U} : \mathtt{m}$ and $\mathtt{U} \to \bullet\mathtt{E} : \mathtt{m}'$ become $\mathtt{E} \to \mathtt{U} : \mathtt{m}$ and $\mathtt{U} \to \mathtt{E} : \mathtt{m}'$, respectively. The attacker can hence impersonate E, inducing the user to generate an `otp` from a given input and steal it when the user inserts it on E. Therefore $\not\vdash MB_{Browser}^S(\sigma)$.

Case $\delta \gg_h$ [F] $\gg_{\gamma'} \delta'$ **where** $\delta \neq \varepsilon$ **and** $\gamma' \neq h$ By definitions of $Actions_k^{in}$ and $Actions_k^{out}$ of Section 3.3.4.1, we know that $\sigma = \ldots; PRE \cdot \ldots \cdot \mathtt{E}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{m} \cdot \mathtt{U}\bullet \rightarrow \bullet\mathtt{A_k} : \mathtt{m} \cdot \ldots \cdot \mathtt{A_k}\bullet \rightarrow \bullet\beta : \mathtt{m}' \cdot \ldots; GOAL$, where $m \in \{\mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A_k}), \mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})), \mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k}))\}$, $\beta \in \{\mathtt{S},\mathtt{E},\mathtt{T}\}$ and $m' \in \{\mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k})), \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A_k})), \mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})\}$.
By definition of MB_e^S, $\mathtt{E}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{m}$ and $\mathtt{A_k}\bullet \rightarrow \bullet\beta : \mathtt{m}'$ become $\mathtt{E} \rightarrow \mathtt{U} : \mathtt{m}$ and $\mathtt{A_k} \rightarrow \bullet\beta : \mathtt{m}'$, respectively. The attacker can hence impersonate E, inducing the user to generate an otp from a given input and collect it by impersonating β. Therefore $\nvdash MB^S_{Browser}(\sigma)$.

Case $\delta \gg_\gamma$ [F] $\gg_h \delta'$ **where** $\delta \neq \varepsilon$ **and** $\gamma \neq h$ By definitions of $Actions_k^{in}$ and $Actions_k^{out}$ of Section 3.3.4.1, we know that $\sigma = \ldots; PRE \cdot \ldots \cdot \beta\bullet \rightarrow \bullet\mathtt{A_k} : \mathtt{m} \cdot \ldots \cdot \mathtt{A_k}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{m}' \cdot \mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{m}' \cdot \mathtt{E} \rightarrow \bullet\mathtt{S} : \mathtt{m}'; GOAL$, where $\beta \in \{\mathtt{S},\mathtt{E},\mathtt{T}\}$ and $m \in \{\mathtt{scrypt}(\mathtt{dK}(\mathtt{S},\mathtt{A_k}), \mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})), \mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k}))\}$ and $m' \in \{\mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k})), \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A_k})), \mathtt{opid}(\mathtt{Operation},\mathtt{UniqueID})\}$.
By definition of MB_e^S, $\beta\bullet \rightarrow \bullet\mathtt{A_k} : \mathtt{m}$ and $\mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{m}'$ become $\beta \rightarrow \bullet\mathtt{A_k} : \mathtt{m}$ and $\mathtt{U} \rightarrow \mathtt{E} : \mathtt{m}'$, respectively. The attacker can hence impersonate β, providing an arbitrary input and get the output. Therefore $\nvdash MB^S_{Browser}(\sigma)$.

Case $\delta \gg_\gamma$ [F] $\gg_{\gamma'} \delta'$ **where** $\delta \neq \varepsilon$ **and** $\gamma, \gamma' \neq h$ This case is symmetric to the one of SE, but only if $e = Browser$.

MM This case is symmetric with the previous one, but for $e = App$

Software Authenticators
In this case, $f_e(\delta \gg_\gamma$ [?] $[\mathrm{F}] \gg_{\gamma'} \delta') =$ ☠.

This implies that $f \in \{$DT, AD, SS, ES, SE, MB, MM$\}$.

We distinguish the following sub-cases.

DT This case is symmetric to DT in the case of hardware authenticators.

AD This attacker, in the case of Software Authenticators, behaves as DT in the case of hardware authenticators,

SS This case is symmetric to SS in the case of hardware authenticators.

ES In this case, we have to specify a set of further distinctions, depending on the input/output of the authenticator.

Case [F] $\gg_h$ **otp** In this case, by definition of $Actions_k^{out}$ given in Section 3.3.4.1, $\sigma = Sem(\text{▦}[\mathrm{F}]\mathtt{otp} \gg_h) = \ldots; PRE \cdot \ldots \cdot \mathtt{A_k}\bullet \rightarrow \bullet\mathtt{U} : \mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k})) \cdot \mathtt{U} \rightarrow \bullet\mathtt{E} : \mathtt{hash}(\mathtt{seed}(\mathtt{S},\mathtt{A_k})) \cdot \ldots; GOAL$.

By definition of ES_e^S, $\mathtt{U} \to \bullet\mathtt{E} : \mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k}))$ become $\mathtt{U} \to \ \mathtt{E} : \mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k}))$.
Since `hash(seed(S,Ak))` has become available to the attacker, $\not\vdash ES_e^S(\sigma)$.

Case $\delta \gg_\gamma$ 📱$^{(?)}$**[K]**$\gg_\gamma \delta'$ **where** $\delta \neq \varepsilon$ **and** $Platform($ 📱$) = Desktop$ In this case, by definition of $Actions_k^{med}$ given in Section 3.3.4.1,
$\sigma = Sem(\delta \gg_\gamma$ 🖩$^{(?)}$[K]$\gg_\gamma \delta') = \ldots; PRE \cdot \ldots \cdot \mathtt{U}\bullet \to \bullet\mathtt{A_k} : \mathtt{fK}(\mathtt{U}, \mathtt{A_k}) \cdot \ldots; GOAL.$
By definition of $\mathrm{ES}_{Browser}^S$, $\mathtt{U}\bullet \to \bullet\mathtt{A_k} : \mathtt{fK}(\mathtt{U}, \mathtt{A_k})$ becomes $\mathtt{U}\bullet \to \mathtt{A_k} : \mathtt{fK}(\mathtt{U}, \mathtt{A_k})$ when $Platform($ 📱$) = Desktop$.
Since `fK(U,Ak)` has become available to the attacker, $\not\vdash ES_{Browser}^S(\sigma)$.

Case $\delta \gg_\gamma$ 📱$^{(?)}$**[K]**$\gg_\gamma \delta'$ **where** $\delta \neq \varepsilon$ **and** $Platform($ 📱$) = Mobile$ This case is symmetric to the previous, but with $e = App$.

SE This case is symmetric to SE in the case of hardware authenticators.

MB This case is symmetric to MB in the case of hardware authenticators.

MM By definitions in Section 3.3.4.1, we can write that $\sigma = Sem(\delta \gg_\gamma$ 📱$^{(?)}$[F]$\gg_\gamma \delta') = \ldots; PRE \cdot \ldots \cdot \beta\bullet \to \bullet\mathtt{A_k} : \mathtt{m} \cdot \ldots \cdot \mathtt{A_k}\bullet \to \bullet\beta' : \mathtt{m}' \cdot \ldots; GOAL$, where $\beta, \beta' \in \{\mathtt{U}, \mathtt{S}, \mathtt{E}, \mathtt{T}\}$ and
$m \in \{\mathtt{scrypt}(\mathtt{dK}(\mathtt{S}, \mathtt{A_k}), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID})), \mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k}))\}$ and
$m' \in \{\mathtt{hash}(\mathtt{seed}(\mathtt{S}, \mathtt{A_k})), \mathtt{crypt}(\mathtt{inv}(\mathtt{pK}(\mathtt{A_k})), \mathtt{opid}(\mathtt{Operation}, \mathtt{UniqueID})\}$.
By definition of MM_e^S, $\beta\bullet \to \bullet\mathtt{A_k} : \mathtt{m}$ and $\mathtt{A_k}\bullet \to \bullet\beta' : \mathtt{m}'$ become $\beta \to \mathtt{A_k} : \mathtt{m}$ and $\mathtt{A_k} \to \beta' : \mathtt{m}'$, respectively, when $Platform($ 📱$) = Mobile$
Thus, $\not\vdash MM_e^S(\sigma)$.

□

Lemma 1. *Given* $P, f_e^1, \ldots, f_e^n$ *s.t.* $f_e^1 \circ \ldots \circ f_e^n(P) =$ ☠ *then* $\forall j \in \{1, \ldots, n\}. \not\vdash f_e^{(j),S}(Sem(f_e^1 \circ \ldots \circ f_e^{j-1} \circ f_e^{j+1} \circ \ldots \circ f_e^n(P)))$

Proof. By definition, we can reorder $f_e^1 \circ \ldots \circ f_e^n$ as $f_e^j \circ f_e^1 \circ \ldots \circ f_e^{j-1} \circ f_e^{j+1} \circ \ldots \circ f_e^n$. Then we define $P' = f_e^1 \circ \ldots \circ f_e^{j-1} \circ f_e^{j+1} \circ \ldots \circ f_e^n(P)$. Thus, we reduced to the statement of Theorem 1.
□

Theorem 2. *Given* $P, f_e^1, \ldots, f_e^n$ *s.t.* $f_e^1 \circ \ldots \circ f_e^n(P) =$ ☠ *then* $\not\vdash f_e^{(1),S}, \ldots, f_e^{(n),S}(Sem(P))$

Proof. We inductively apply Lemma 1 to $f_e^1, \ldots, f_e^n$. □

As for Section 4.1.1, we now apply the attacker combination SS∘DT on Nordea1. In this case, however, we use its SPS counterpart, namely $\mathrm{SS}_e^S \circ \mathrm{DT}_e^S$. The following example shows that, as SS∘DT compromised Nordea1 ($\mathrm{SS} \circ \mathrm{DT}(\text{Nordea1}) =$ ☠), $\mathrm{SS}_e^S \circ \mathrm{DT}_e^S$ transforms the SPS specification of Nordea1 in a way that the authentication property is violated ($\not\vdash \mathrm{SS}_e^S \circ \mathrm{DT}_e^S(Sem(\text{Nordea1}))$).

Example 11. *Let us consider Nordea1. As previously explained in Section 4.1.1, SS∘DT(Nordea1)* = ☠.

We now calculate $SS_e^S \circ DT_e^S(Sem(Nordea1))$*.To do so, we calculate* $SS_e^S \circ DT_e^S(Sem(🔑))$ *and* $SS_e^S \circ DT_e^S(Sem(\texttt{opid} \gg_h 🖩[O,K] \gg_h \texttt{otp}))$ *separately.*

In the first case, $\sigma = Sem(🔑) = \ldots; PRE \cdot \texttt{U} \rightarrow \bullet\texttt{E} : \texttt{mem}(\texttt{U},\texttt{S},1) \cdot \texttt{E} \rightarrow \bullet\texttt{S} : \texttt{mem}(\texttt{U},\texttt{S},1)$; *GOAL. Due to the capabilities of* $SS_e^S \circ DT_e^S$, $\texttt{U} \rightarrow \bullet\texttt{E} : \texttt{mem}(\texttt{U},\texttt{S},1)$ *becomes* $\texttt{U} \rightarrow \texttt{E} : \texttt{mem}(\texttt{U},\texttt{S},1)$. *Therefore,* `mem(U, S, 1)` *can be compromised, meaning that* $\not\vdash SS_e^S \circ DT_e^S(\sigma)$.

In the second case, $\sigma = Sem(🔑) = \ldots; PRE \cdot \texttt{S} : \texttt{Number UniqueID} \cdot \texttt{S}\bullet \rightarrow \bullet\texttt{E} :$
$\texttt{scrypt}(\texttt{dK}(\texttt{S},\texttt{A}_2),\texttt{opid}(\texttt{Operation},\texttt{UniqueID}))\cdot$
$\texttt{E}\bullet \rightarrow \bullet\texttt{U} : \texttt{scrypt}(\texttt{dK}(\texttt{S},\texttt{A}_2),\texttt{opid}(\texttt{Operation},\texttt{UniqueID}))\cdot$
$\texttt{U}\bullet \rightarrow \bullet\texttt{A}_2 : \texttt{scrypt}(\texttt{dK}(\texttt{S},\texttt{A}_2),\texttt{opid}(\texttt{Operation},\texttt{UniqueID})))\cdot$
$\texttt{U}\bullet \rightarrow \bullet\texttt{A}_2 : \texttt{fK}(\texttt{U},\texttt{A}_2)\cdot$
$\texttt{A}_2\bullet \rightarrow \bullet\texttt{U} : \texttt{crypt}(\texttt{inv}(\texttt{pK}(\texttt{A}_2)),\texttt{opid}(\texttt{Operation},\texttt{UniqueID}))\cdot$
$\texttt{U} \rightarrow \bullet\texttt{E} : \texttt{crypt}(\texttt{inv}(\texttt{pK}(\texttt{A}_2)),\texttt{opid}(\texttt{Operation},\texttt{UniqueID}))\cdot$
$\texttt{E} \rightarrow \bullet\texttt{S} : \texttt{crypt}(\texttt{inv}(\texttt{pK}(\texttt{A}_2)),\texttt{opid}(\texttt{Operation},\texttt{UniqueID}))$; *GOAL.*

Due to the capabilities of $SS_e^S \circ DT_e^S$,
$\texttt{E}\bullet \rightarrow \bullet\texttt{U} : \texttt{scrypt}(\texttt{dK}(\texttt{S},\texttt{A}_2),\texttt{opid}(\texttt{Operation},\texttt{UniqueID}))$,
$\texttt{U}\bullet \rightarrow \bullet\texttt{A}_2 : \texttt{scrypt}(\texttt{dK}(\texttt{A}_2),\texttt{opid}(\texttt{Operation},\texttt{UniqueID})))$,
$\texttt{U}\bullet \rightarrow \bullet\texttt{A}_2 : \texttt{fK}(\texttt{U},\texttt{A}_2)$,
$\texttt{A}_2\bullet \rightarrow \bullet\texttt{U} : \texttt{crypt}(\texttt{inv}(\texttt{pK}(\texttt{A}_2)),\texttt{opid}(\texttt{Operation},\texttt{UniqueID}))$ *and*
$\texttt{U} \rightarrow \bullet\texttt{E} : \texttt{crypt}(\texttt{inv}(\texttt{pK}(\texttt{A}_2)),\texttt{opid}(\texttt{Operation},\texttt{UniqueID}))$
become
$\texttt{E}\bullet \rightarrow \texttt{U} : \texttt{scrypt}(\texttt{dK}(\texttt{S},\texttt{A}_2),\texttt{opid}(\texttt{Operation},\texttt{UniqueID}))$,
$\texttt{U} \rightarrow \texttt{A}_2 : \texttt{scrypt}(\texttt{dK}(\texttt{A}_2),\texttt{opid}(\texttt{Operation},\texttt{UniqueID})))$,
$\texttt{U} \rightarrow \texttt{A}_2 : \texttt{fK}(\texttt{U},\texttt{A}_2)$,
$\texttt{A}_2\bullet \rightarrow \texttt{U} : \texttt{crypt}(\texttt{inv}(\texttt{pK}(\texttt{A}_2)),\texttt{opid}(\texttt{Operation},\texttt{UniqueID}))$ *and*
$\texttt{U} \rightarrow \texttt{E} : \texttt{crypt}(\texttt{inv}(\texttt{pK}(\texttt{A}_2)),\texttt{opid}(\texttt{Operation},\texttt{UniqueID}))$.

Therefore, the attacker can steal `fK(U, A2)` *and hijack a session with* $\mathbf{A}_2$*, meaning that* $\not\vdash SS_e^S \circ DT_e^S(\sigma)$.

Thus, $\not\vdash SS_e^S \circ DT_e^S(Sem(Nordea1))$.

The SPS obtained after the attack is reported in Listing 4.1. The symbol ⊘ *denotes that a security property has been dropped due to the effect of* $SS_e^S \circ DT_e^S$.

□

```
1  Types:
2          Agents U, E, S, A2;
3
4  Mappings:
5          mem : Agent,Agent,Number -> Msg;
6          pK: Agent -> PublicKey;
7          inv: PublicKey -> PrivateKey;
8          fK: Agent,Agent -> SymmetricKey;
9          dK: Agent,Agent -> SymmetricKey;
10
11 Formats:
12         opid(Msg,Number);
13
14 Knowledge:
15         U: U, E, A2, mem(S,U,1), pk(A2), fK(U,A2);
16         E: E, U, S, pK(A2);
17         S: S, U, E, mem(S,U,1), pK(A2), dK(S,A2);
18         A2: U, E, A2, dK(S,A2), fK(U,A2), pk(A2), inv(pK(A2));
19
20 Actions:
21         U: Msg Operation
22         U →⊘ E: Operation
23         E →● S: Operation
24         U →⊘ E: mem(S,U,1)
25         E →● S: mem(S,U,1)
26         S : Number UniqueID
27         S ●→● E: scrypt(dK(S,A2),opid(Operation,UniqueID))
28         E ●→⊘ U: scrypt(dK(S,A2),opid(Operation,UniqueID))
29         U ⊘→⊘ A2: scrypt(dK(A2),opid(Operation,UniqueID)))
30         U ⊘→⊘ A2: fK(U,A2)
31         A2 ●→⊘ U: crypt(inv(pK(A2)),opid(Operation,UniqueID))
32         U →⊘ E: crypt(inv(pK(A2)),opid(Operation,UniqueID))
33         E →● S: crypt(inv(pK(A2)),opid(Operation,UniqueID))
34
35 Goals:
36         S authenticates U over Operation
```

Listing 4.1: SPS specification of Nordea1 after SS_e^S and DT_e^S attacks.

4.2 Compliance

Our framework allows to evaluate an MFA implementation in terms of compliance with a set of requirements and best practices. The evaluation can be done by focusing either on the MFA protocol itself or on other features characterizing the MFA implementation, such as the enrollment and binding phases or the adoption of exemptions. This is possible because the published laws

and guidelines must not impose the employment of a particular product or technology. Therefore, in the majority of cases, the presented requirements or definitions do not specify details on the internal functionality of the entities that are involved in MFA.

The rationale behind this kind of analysis is simple. The first step is to extract the most relevant aspects of a document (e.g., a directive), that should characterize the design of an MFA protocol (or implementation) to be evaluated, e.g., "the usage of SMS an phone calls should be avoided". Then, for each of the extracted requirements, it is necessary to define a criterion for establishing if a certain MFA protocol (or implementation) complies with it. Clearly, this criterion should refer to some feature that can be expressed with our language, e.g., "$\blacksquare \notin P$". The analysis will hence be performed by matching the MFA protocol (or implementation) with the criteria and evaluating the compliance with each of the defined requirements.

In this research work, we extracted and encoded a set of requirements deriving from EBA regulations ([Eur13a, Eur13b, Eur15, Eur17] and a set of best practices deriving from [NIS17, PCI16a, Pin09, Gem15, Cen16]. Then, we used it for evaluating the MFA implementations of several banks around the world (see Section 6.2). However, such requirements and best practices can reasonably be used for evaluating any MFA implementation.

In the following section, we describe the process of extraction and encoding of both the security requirements and best practices, together with the criteria for evaluating how an MFA protocol or an implementation comply with them.

4.2.1 Security Requirements

During our research on the Online Banking use case, a set of official documents, guidelines and laws concerning the employment of MFA solutions in the online banking context has been analyzed. From this study, we derived a set of requirements, representing the key features that have to be included (according to the main authorities in the ebanking context) in order to provide a proper security level.

The main source of requirements for MFA-based e-payment systems are EU regulations.[4] We group requirements according to their scope:

(i) authenticator requirements refer to specific features that authenticators must comply with; *(ii) digital authentication* requirements refer to properties of the MFA protocols employed for digital authentication; *(iii) identity proofing* and *binding* requirements refer to properties of enrollment

[4]We also considered the Payment Card Industry (PCI) Data Security Standard [PCI16a, PCI16b] (becoming effective in 2018). As a matter of fact, PCI applies to any service that stores and manages payment card data, which includes e-banking services. Nevertheless, the requirements specified in the Data Security Standard are either too generic (e.g., provide documentation to the user for an informed usage of MFA) or too specific for the single implementation (that we are not able to test). For this reason, these requirements have not been considered in our survey.

Table 4.13: Key Requirements in EU regulations and directives.

		RSIP [Eur13a]	**RSMP** [Eur13b]	**PSD2** [Eur15]	**RTS** [Eur17]
Authenticators					
RL1	If a software authenticator or an authentication code is used through a multi-purpose device, the integrity of the device must be checked	○	○	○	●
Digital Authentication					
RL2	MFA protocols must be always employed when the user performs risky operations	◑	◑	●	●
RL3	Every MFA protocol must employ at least two different types of AFs	●	●	●	●
RL4	Every MFA protocol must employ at least two independent AFs	◑	◑	●	●
RL5	Every MFA protocol must result in the generation of an authentication code that is unique, dynamically linked to a specific operation and accepted only once.	○	○	◑	●
RL6	Every MFA protocol must make the user aware of crucial information on the operation she is going to authorize	○	○	○	●
Enrollment and Binding					
RL7	Identity proofing must be performed with a high level of confidence	○	○	○	●
RL8	The binding procedure for every authenticator must be executed in a secure manner	●	●	◑	●
RL9	Every remotely delivered authenticator must be activated before its usage	○	○	○	●

and binding phases, respectively. Below we discuss those requirements, which are summarized in Table 4.13.

4.2.1.1 Authenticator requirements

RTS [Eur17] requires that authenticators are tamper-proof. This, however, is a hard requirement to meet, in case of software authenticators that run on multi-purpose, e.g., mobile devices (this will be detailed in the following sections). As highlighted in [HM18], software authenticators are relatively easy to compromise if the platform running them is compromised. Therefore, according to [Eur17], when a software authenticator or an authentication code is used through a multi-purpose device, such a device must be checked against alterations (**RL1**). The aim of this requirement is hence to limit the aforementioned weakness as much as possible, by imposing banks to equip their software authenticators and applications with mechanisms for ensuring the integrity of mobile devices and potentially blocking the execution of payments from an insecure endpoint.

4.2.1.2 Digital Authentication requirements

The specifications concerning digital authentication based on MFA have evolved over time. While some initial requirements on MFA were given in RSIP [Eur13a], more precise and stringent requirements were defined in later regulations and directives. For instance, PSD2 [Eur15] introduces the concept of authentication code and dynamic linking (further refined in RTS [Eur17]).

EBA defines specific situations in which MFA must be adopted (**RL2**). In particular, PSD2 and RTS require the adoption of MFA when (i) accessing a personal account for the first time in 90 days, (ii) initiating a payment transaction and (iii) executing any operation that may imply a risk of fraud or other abuses. Hereafter, we refer to such operations as *risky operations*. Intuitively, a risky operation is an operation that may lead to a leakage of sensitive data or to an economical damage. The operations to be considered as *risky* were identified by EBA after public consultations with several stakeholders within the online banking context. Specifically, the identification process took into account the outcome of empirical research and risk analysis procedures that are normally adopted in the field.

EBA also explicitly lists a number of situations in which MFA may not be employed. These exemptions for the MFA employment are regulated in [Eur17]. In particular, operations such as (i) checking the account balance, (ii) paying a trusted beneficiary, (iii) executing a recurring transaction and (iv) executing a bank transfer between user's own accounts may not require MFA. This requirement has an impact on both the security and usability of MFA protocols. In particular, **RL2** helps in identifying the situations in which the security level provided by an MFA protocol should be reasonably high. The exemptions, instead, helps identifying those situations in which the user can be relieved from MFA, increasing the usability and perceived ease of use.

Further requirements are provided for the design of the MFA protocol itself. A first design requirement, defined already in [Eur13a], concerns the variety of AFs employed in a MFA protocol. In particular, EBA requires that AFs are *distinct*, i.e., at least two distinct types of AFs (e.g., one knowledge and one ownership factor) should be employed in the verification process (**RL3**). This requirement is crucial for MFA protocols, since the variety of AFs can strongly increase their security. The differentiation of AFs, indeed, forces a potential malicious agent to adopt different techniques in order to compromise an identity proof, hence making it difficult for the attacker to execute an MFA protocol on behalf of the user.

Another fundamental design requirement for MFA protocols is the *independence* of the employed AFs. Specifically, after compromising one AF, an attacker must have no advantage for compromising the others. This concept is formalized in [Eur13a, Eur15, Eur17], where it is stated that AFs adopted by an MFA protocol must be independent (**RL4**). Reasonably, to compromise an AF, the attackers must control the authenticator that attests it. Therefore, an attacker has to control at least two different authenticators to compromise two AFs. Furthermore, EBA requires that an MFA protocol "shall result in the generation of an authentication code" [Eur17]. EBA also states that the authentication code is the information used to authorize a specific payment

transaction. For this reason, an authentication code should be OTP-generated (through a device or other cryptographic facility) and must be uniquely linked to a specific transaction (**RL5**). This requirement aims to improve the security level of MFA by strictly linking an authentication code to its context of use; accordingly, even if intercepted by a malicious agent, it can only be used for an operation that has been previously confirmed.

Finally, EBA requires in [Eur17] that the user is made aware of (the crucial information about) the ongoing transaction (**RL6**). The information provided to the user should include (i) the amount of the transaction, (ii) the payee and (iii) the generated codes. Intuitively, this requirement aims to reduce the risk for the user to be induced to perform unwanted actions.

4.2.1.3 Enrollment and Binding requirements

Regulations and directives also define requirements on both the enrollment and binding procedures. For the former, EBA requires that users are identified with a high level of assurance (**RL7**). This is because issues in this phase can affect the reliability of the entire MFA process. In particular, EBA requires that user identification is carried out by a trained person [Eur17]. Additional constraints are imposed by eIDAS regulations [Eur14], which require user identification to be based on highly reliable identification proofs. For the binding process instead, two main requirements are defined by the EBA. The first addresses the delivery of authenticators [Eur13a, Eur13b]. In particular, each authenticator must be delivered exactly to the intended user and in a secure manner (**RL8**). Accordingly, authenticators should be delivered to users personally after an in-person (*de visu*) identification. In case of "remote binding", authenticators should be delivered only after the user has been identified through MFA (by means of previously bounded authenticators).

An additional (and more specific) requirement concerning the binding phase requires an activation procedure for those authenticators that have been remotely delivered (**RL9**). Leveraging an activation procedure, indeed, a service provider can provide users with a unique code to be inserted in the delivered authenticator in order to uniquely bind it to their identity.

The aforementioned requirements (**RL7**, **RL8** and **RL9**) are clearly intended to limit the possible risks in the operations that are preliminary for the MFA usage. The security guarantees offered by an MFA protocol do indeed depend on the aforementioned operations: if an attacker manages to obtain an authenticator, the overall security of any MFA protocol depending on that authenticator is compromised.

4.2.1.4 Evaluation Criteria for Security Requirements

The criteria for evaluating the considered banks in terms of the security requirements are defined as follows.

Table 4.14: Summary of the evaluation criteria for requirements.

ID	★	⯪	☆	**Criteria**
RL1	Every	Some	No	banking application (including software authenticators) provides a device integrity check
RL2	No	n.a.	Some	risky operation is performed without MFA
RL3	Every	Some	No	MFA protocol relies on at least two AF
RL4	Every	Some	No	MFA protocol relies on at least two authenticators
RL5	Every	Some	No	MFA protocol contains `opid`» · »`otp` or `otp`» · »`otp`
RL6	Every	Some	No	MFA protocol uses at least one of [...] and [...]
RL7			n.a.	used for the enrollment
RL8	Every	Some	No	binding includes or
RL9	Every	Some	No	binding is (·, , ·), (·, ·,) or (·, ·,)

It is worth noting that some requirements are defined at the level of banking application, authenticator and MFA protocol (e.g., **RL1**, **RL3** to **RL6**, **RL8** and **RL9**) and others at the level of bank. In our analysis, we evaluate the former group of requirements also at the level of bank. In this case, we consider three possible levels of fulfillment defined through the quantification over the banking application, authenticator and MFA protocols adopted by a bank: *fulfilled* (★) denotes that all adopted target elements (banking applications, authenticators or MFA protocols) satisfy the requirement; *partially/possibly*[5] *violated* (⯪) denotes that some (but not all) adopted target elements satisfy the requirement; and *violated* (☆) denotes that none of the target elements satisfies the requirement. The criteria for assessing the compliance with requirements and best practices are summarized in Table 4.14 and Table 4.16, respectively.

RL1. To determine whether a bank meets this requirement, we verify its software authenticators and mobile banking applications.[6] In particular, we inspect the code, looking for the use of root detection mechanisms. Thus, we state that **RL1** is fulfilled when the code of every application includes these checks, partially violated when only some applications include these checks, and violated otherwise.

RL2. This requirement focuses on the protection of risky operations (see Section 4.2.1) with MFA. Trivially, **RL2** is satisfied by a bank if the bank does not allow the execution of risky operations without MFA. Otherwise, **RL2** is violated.

RL3. This requirement is fulfilled by a bank if and only if all employed MFA protocols involve at least two AFs of different type. For instance, a protocol in which is combined with [O] complies with **RL3** (since subsumes a knowledge factor), while a protocol using authenticators and [O] does not (as both authenticators assert an ownership factor). If only a subset of the MFA protocols offered by a bank meets this criterion, the requirement is considered partially violated by the bank. Finally, if none of the offered MFA protocols involve at least two AFs of

[5]The compliance of a bank with some requirements (e.g., **RL7**) depends on the specific implementation adopted by the bank. In this cases, ★ indicates that the implementation adopted by a bank can potentially violate the requirement.

[6]We only considered the Android version of banking applications.

different type, **RL3** is violated.

RL4. As stated in the previous section, we assume that two AFs are independent when an adversary has to control at least two authenticators to compromise both. For instance, this is the case for 🔑;▦. Conversely, two AFs are not independent when they are attested by a single, multi-factor authenticator, e.g., 🖩[O,K]. Under this interpretation, **RL4** requires that every MFA protocol adopted by a bank employs at least two distinct authenticators. Hence, a bank partially violates **RL4** if some of the MFA protocols it offers do not use more than one authenticator; if none of its MFA protocols uses more than one authenticator, we consider **RL4** violated.

RL5. This requirement is satisfied by a bank if and only if all employed MFA protocols result in an authenticator output that is uniquely associated to the ongoing operation. In symbols, we require that an MFA protocol includes at least one authenticator with the form `opid` $\gg \cdot \gg$ `otp`[7] or `otp`$\gg \cdot \gg$ `otp`. For example, an MFA protocol relying on `opid`$\gg_i$ 🔐[O,K] $\gg_i$ `otp` does link the output to the ongoing operation, while one only relying on 🖩[O,K] $\gg_h$ `otp` does not. If only some of the protocols satisfy it, **RL5** is considered partially violated and, if none of the protocols returns the desired authenticator output, **RL5** is violated.

RL6. This requirement is satisfied by a bank if and only if all employed MFA protocols employ at least one authenticator labeled with **?**, i.e., in the case of 🖩[...] and 🔐[...]. In contrast, if such an authenticator is only used in some of the MFA protocols, **RL6** is partially violated; if none of the provided MFA protocols uses it, **RL6** is violated.

RL7. The required level of assurance is clearly achieved when enrollment is carried out in person (🏢). Otherwise, i.e., when enrollment is performed remotely (🌍), we claim that **RL7** is possibly violated.

RL8. As for **RL7**, the binding procedure for an authenticator provides the required level of assurance only if one step is done at the bank (🏢) or through an MFA protocol (🔒). Note that an MFA protocol executed in a binding step should comply at least with **RL3** and **RL4** and the authenticators it employs should have been bound in compliance with **RL8** and **RL9**. If the request step includes **E**, it is necessary to consider the modality in which the enrollment is executed (for that bank). Thus, a bank satisfies **RL8** if the binding procedure for all its authenticators provide the required level of assurance. Instead, the requirement is partially violated when only some binding procedures rely on in-person identification or MFA. Finally, if no binding procedure requires in-person identification or MFA, **RL8** is violated.

RL9. This requirement is satisfied under two circumstances: either the delivery phase consists of 🏢 or there is a secure activation step (i.e., 🏢 or 🔒). On the other hand, we consider **RL9** possibly violated if these conditions are met by a bank only for some authenticators. In all other cases, the bank violates **RL9**.

[7] We assume that if an `opid` is received as an input, it is actually used to generate the output, which is therefore assumed unique. Unfortunately, we cannot verify if the output is accepted only once by the server.

Table 4.15: Selected Best Practices.

		NIST [NIS17]	PCI-SSC [PCI17]	CENTRIFY [CEN16]	GEMALTO [GEM15]	PING IDENTITY [PIN09]
Authenticators						
BP1	A software authenticator should be integrated in the mobile banking application (if any)	○	○	○	○	●
Digital Authentication						
BP2	MFA protocols should rely on standard solutions	○	◑	●	●	○
BP3	Step-up authentication should be adopted	○	◑	●	●	●
BP4	MFA protocols should limit SMS reception as much as possible	●	●	○	○	●
Enrollment and Binding						
BP5	Identity proofing should be executed with high level of confidence	●	◑	○	○	○
BP6	The binding procedure should be executed in a secure manner	●	●	○	●	●
BP7	Two authenticators attesting ownership factors should be bound after the enrollment	●	○	○	○	○
BP8	The user should be offered with multiple authenticators of different types	●	○	●	◑	●

4.2.2 Best Practices

By gathering the guidelines released by NIST [NIS17] and PCI-SSC (MFA Guidelines, [PCI17]) as well as by three independent vendors of digital identity security systems, i.e., Gemalto [Gem15], PingIdentity [Pin09] and Centrify [Cen16], we have categorized a set of *best practices*. We categorize best practices following the same criteria used for the requirements.

4.2.2.1 Authenticator best practices

Best practices on authenticators put a major emphasis on mobile devices, e.g., smartphones and tablets. For instance, Ping Identity [Pin09] states that, if a service requiring MFA employs a dedicated mobile application, the capabilities of the authenticator should be integrated in it (**BP1**). That is, the service provider should not use a separate authentication application. This best practice aims to improve the usability of MFA protocols by reducing the amount of applications that a user has to interact with. **BP1** can somehow be related to **RL1**: joining two software applications into one might help in the security measures effectiveness, since security functionalities do not have to be implemented twice.

4.2.2.2 Digital authentication best practices

These best practices deal with the digital authentication process with a particular emphasis on MFA protocols and their features. For example, Centrify [Cen16] and Gemalto [Gem15] recommend to implement MFA protocols using standard solutions (**BP2**). This because, unlike proprietary algorithms, standardized algorithms go through public scrutiny by industry and security experts, which reduces the chance of any inherent weakness or vulnerability. This best practice adds some constraints to the design requirements related to MFA protocols (i.e., **RL3** to **RL6**) by encouraging the employment of solutions that comply with the aforementioned requirements and by limiting the adoption of ad-hoc solutions that may introduce security issues.

Moreover, best practices in [Cen16, Gem15, Pin09] provide recommendation on the situations in which MFA should be adopted. In particular, they recommend to implement adaptive or "step-up" authentication procedures. This means that MFA should be avoided when not strictly necessary (**BP3**). A way to implement this is to require an increasing number of authenticators depending on the actions to be performed by the user. This best practice aims to improve the perceived ease-of-use during the access to remote data. Step-up authentication relieves users from the burden to achieve an unnecessary high security level by requiring MFA only in situations where it is needed. It is worth noting that this best practice exemplifies the exemptions introduced in **RL2** by suggesting practical solutions to be adopted in low-risk scenarios.

Another best practice targets the adoption of SMS services, therefore adding constraints to MFA protocol requirements. In particular, both NIST and PCI-CSS (in [NIS17] and [PCI17], respectively) deprecate the usage of out-of-band authentication via SMS (**BP4**). This is because the reliability of this kind of authenticator has been recently questioned: a large amount of malware has specifically targeted this authentication method to obtain sensitive data for MFA [Vir14, Kas13]. Moreover, the advantage of using alternative channels (compared to classic HTTPS connection) may be nullified if an SMS is received on the same phone from which a payment operation is started [PCI17]. According to [Hag07], the adoption of SMS can lead to situations in which the mobile phone of the user constitutes a single point of failure whose exploitation can compromise the security of all communication channels.

4.2.2.3 Enrollment and binding best practices

NIST provides a detailed explanation in [NIS17] on how enrollment and, in particular, identity proofing should be performed. In our context, it can be summarized by asserting that identity proofing should be performed with high level of confidence (**BP5**). This can be performed by an in-person identification and/or based on documents and certifications issued by a so-called qualified entity (e.g., a national authority). Regarding the binding process, the best practices released by many public and private authorities [NIS17, PCI17, Gem15, Pin09] require service providers to bind authenticators to the user's identity in a secure manner (**BP6**). This because

Table 4.16: Summary of the evaluation criteria for best practices.

ID	★	⯪	☆	**Criteria**
BP1	Every	n.a.	No	and/or is integrated in MP endpoint
BP2	Every	Some	No	relevant (cfr. Table 4.17) API is used by dedicated mobile applications
BP3	✔or ✔✔	n.a.	✖	exemptions are adopted by the online service
BP4	No	Some	Every	MFA protocol relies on
BP5	–	–	–	*same as **RL7***
BP6	–	–	–	*same as **RL8***
BP7	2+	1	0	among [$\{O\}\cup$F], [$\{O\}\cup$F] and have binding procedure of the form ($\mathbf{E}$,·,·)
BP8	2+	n.a.	1	type of authenticator is provided

"an authentication mechanism is only as strong as the binding process that issued the credentials" [Pin09]. It is worth noting that **BP5** and **BP6** are strongly related to requirements **RL7** and **RL8**, respectively. This underlines the paramount importance of the identity proofing and binding procedures in the security of MFA systems.

According to [NIS17], at least two physical authenticators should be bound to the user's identity immediately after enrollment (**BP7**). This best practice is related to **RL9** and permits users to be immediately able to provide ownership factors for digital authentication or for future remote bindings. Moreover, a number of authorities [NIS17, Cen16, Pin09] recommend service providers to support flexible authentication procedures (**BP8**). This reduces to allowing the user to select among multiple, alternative authenticators of different types. Customizing the MFA experience could, indeed, increase the perceived usability of an MFA implementation. Binding multiple authenticators also makes it possible to recover from the loss or theft of other authenticators that a user might posses.

The best practices above have been extracted from guidelines and white papers concerning MFA applied in a generic context. Nevertheless, they can be easily mapped to one or more requirements defined in EBA documents. Differently from what one may expect, the best practices related to the security aspects of MFA are rarely more strict than the related requirements (except for the one concerning the SMS usage). On the other hand, it is noticeable how their adoption can increase the ease-of-use of MFA protocols.

4.2.2.4 Evaluation Criteria of Best Practices

The criteria for evaluating an online service in terms of the presented best practices are defined as follows.

BP1. This best practice is fulfilled whenever an online service offering software authenticators integrates their functionality in the dedicated mobile application. Otherwise, the best practice is violated. In the case an online service does not provide users with any software authenticator (

Table 4.17: Relationship between MFA protocol elements and relevant Android APIs.

Element	Relevant API	Used for
always	`com.google.android.gms.safetynet.*`	Integrity
I	`android.hardware.fingerprint.*`	Fingerprint
O	`android.security.keystore.*`	Secure storage
K	`android.widget.EditText` (`"textPassword"` mode)	Password field
📶	`android.telephony.TelephonyManager`	Telephony network
📱	`firebase.messaging.*`	Push notification
$\gg_i$	`android.content.Intent` (permission protected)	Android IPC
$\gg_o$	`gms.vision.barcode.*` or `firebase.ml.vision.*`	Optical code
$\gg_n$	`javax.net.ssl.*`	HTTPS & SSL

and 📱), **BP1** is considered to be fulfilled.

BP2. In general, checking the adoption of standard technologies requires disassembling and inspecting the target object. Since we cannot do this for remote services and authenticator devices, here we only focus on software authenticators (used both for IP and MP) and dedicated mobile applications. In particular, we decompiled the Android applications (both software authenticators and dedicated applications) offered by each an online service and inspected the code looking for standard APIs (according to [Anda, Andb, Andc]). Table 4.17 defines the relationship between the (elements of the) MFA protocols and the standard APIs that it should use, namely the relevant APIs. Thus, we claim that an online service satisfies **BP2** when all its Android applications use the standard APIs that are relevant for the MFA protocols they are involved in. For instance, for the MFA protocol `opid` $\gg_n$ 📱[O] $\gg_n$ `otp` we expect to find the APIs `com.google.android.gms.safetynet.*` (always), `javax.net.ssl.*` (due to `opid` $\gg_n$ and $\gg_n$ `otp`), `firebase.messaging.*` (due to 📱) and `android.security.keystore.*` (due to [O]). If only some applications use those API, we say that **BP2** is possibly violated and, when no application uses them, we say that the best practice is violated. Note that this evaluation criterion adheres as much as possible to **BP2**. Nonetheless, we stress that the usage of commercial APIs – different from the ones listed above – does not necessarily mean that a less secure digital authentication is provided. However, a security analysis of APIs is out of the scope of this work; the implications of this criterion on our analysis are discussed in Section 6.3.

BP3. This best practice is fulfilled when an online service adopts a step-up authentication driven by the risk level of the operation to be performed. In this context, the fact that an online service defines multiple risk levels is indicated by the exemptions it adopts. Thus, we state that an online service satisfies **BP3** if it adopts some exemption, i.e., either ✔✔ or ✔. We consider **BP3** violated by online services that do not use exemptions, i.e., ✖.

BP4. This best practices concerns the usage of SMS messages (received through 📶) in MFA

protocols. Clearly, if none of the MFA protocols employed by an online service uses , the online service satisfies **BP4**. Otherwise, if some (but not all) of its MFA protocols leverage , the online service partially violates **BP4**; if all MFA protocols make use of , the online service violates **BP4**.

BP5. This best practice requires users to exhibit an official identification document to a clerk. The best practice is hence subsumed by **RL7**. Therefore, we evaluate **BF5** using the same criterion defined for **RL7**.

BP6. This best practice is subsumed by **RL8**. Accordingly, its satisfaction is assessed using the criterion defined for **RL8**.

BP7. This best practice states that the user should receive at least two authenticators devices (or one authenticator device and a look-up secret) immediately after the enrollment phase. Thus, an online service satisfies **BP7** if the binding procedure for at least two authenticator devices (or look-up secrets) is performed during enrollment (**E**), i.e., the procedure has the form (**E**,·,·). The best practice is possibly violated if this happens for only one authenticator. Otherwise, **BP7** is violated.

BP8. This best practice is fulfilled by an online service whenever it provides its users with at least two different types of authenticators, e.g., and . Otherwise, if at most one type is provided, **BP8** is violated.

4.3 Complexity

A key aspect for the adoption of MFA protocols is their ease-to-use [CDFN13, WDRJ10, Alt16]. Indeed, the users might be discouraged to execute a cumbersome or complex protocol. Moreover, if a protocol is too complex and hard to follow, a user might incur in errors either spontaneous or, even worse, induced by an attacker.

A standard approach [ISO18] for evaluating the usability of a system is to measure its *effectiveness* (i.e., the accuracy and completeness with which users achieve specified goals), *efficiency* (i.e., the resources used in relation to the results achieved) and *satisfaction* (i.e., how the system meets the user expectations) related to reaching of a given goal. A number of studies [WDRJ10, Alt16, WDCJ09] have applied those measures to analyze MFA protocols and, in general, solutions for digital authentication. For instance, Weir et al. [WDCJ09] applied them for the analysis of two-factor authentication protocols where effectiveness was assessed by checking task completion records and usage of help, efficiency by counting the time needed to complete the authentication process and satisfaction by questioning users immediately after they authenticated. The same usability metrics were used in [Alt16], where a broader scope of MFA protocols was investigated. Other studies [CDFN13, KPCS15] focus on user satisfaction and, in general, perceived usability of MFA protocols for online banking. These studies apply the System Us-

ability Scale (SUS) [Bro96], which relies on a predefined questionnaire to provide a subjective measure of the usability perceived by users about a target system.

In our study, we focus on the efficiency and, in particular, on the complexity of MFA protocols as several studies [CDFN13, KPCS15, WDRJ10] recognized the complexity of MFA protocols as a critical aspect in the adoption of those protocols. In particular, we define a metric to evaluate the complexity of an MFA protocol (i.e., how much a protocol is difficult to use) that measures the amount of "resources" necessary to execute an MFA protocol (e.g., remembering a password, bringing a device). We consider three main types of resources for our evaluation, namely *memory*, *(manual) operations* and *(extra) devices*. The first type is used to determine the number of knowledge factors used in a MFA protocol (i.e., 🔑 and [K]), the second the number of input/output operations that have to be carried out by the user ($\gg_h$) and the third the number of (non general-purpose) devices that have to be carried by the user (i.e., ▦ and 🖩). For example, consider the MFA protocol 🔑 ; `opid` $\gg_h$ 🖩[O,K] $\gg_h$ `otp`. Its memory, operations and devices values are 2, 2 and 1, respectively. The overall complexity value (hereafter called *complexity score*) is obtained as the sum of these three numbers, e.g., 5 in the previous example.

Chapter 5

Framework Implementation

The framework presented in the previous chapter has been implemented in a working prototype. Such prototype is called Multi-Factor Authentication Specification and Analysis tool (MuFASA). This tool allows to describe an MFA protocol design leveraging a user-friendly interface, generate the corresponding specification in SLaMP and analyze it.

MuFASA is based on the modeling and analysis process schematically depicted in Figure 5.1. Our working assumption is to obtain a description of the MFA protocol from the experience of common users, acquired by running the MFA protocol (*usage* phase). Therefore, the user knowledge has to be translated into a model of the MFA protocol behavior (*translation* phase). We rely on a questionnaire to support the modeling process for users with no technical skills. The users fill the questionnaire and provide a description of the MFA protocols they use. The compiled questionnaire is then automatically processed (*modeling* phase), obtaining the corresponding model of the MFA protocol, specified in SLaMP.

The model then passes through an *analysis* phase, in which it is *a)* validated against a set of built-in adversaries, *b)* validated against a list of specifications and *c)* evaluated in terms of usability. The adversaries consist in the threat models presented in Section 4.1.1. Instead, the specifications consist of a collection of requirements and best practices, presented in Section 4.2.1 and 4.2.2, respectively released. Finally, the ease-of-use of the protocol is measured in terms of complexity (described in Section 4.3).

The final result of the process is a report, i.e., the output of the *reporting* phase. The report contains the results of the security analysis as well as other metrics of interest. In particular, we provide (i) a risk profile (in terms of the adversaries that might compromise the protocol), (ii) a compliance checklist (i.e., what are the requirements and guidelines that the protocol meets) and (iii) a complexity score (i.e., how many operations must be correctly executed by the user to authenticate).

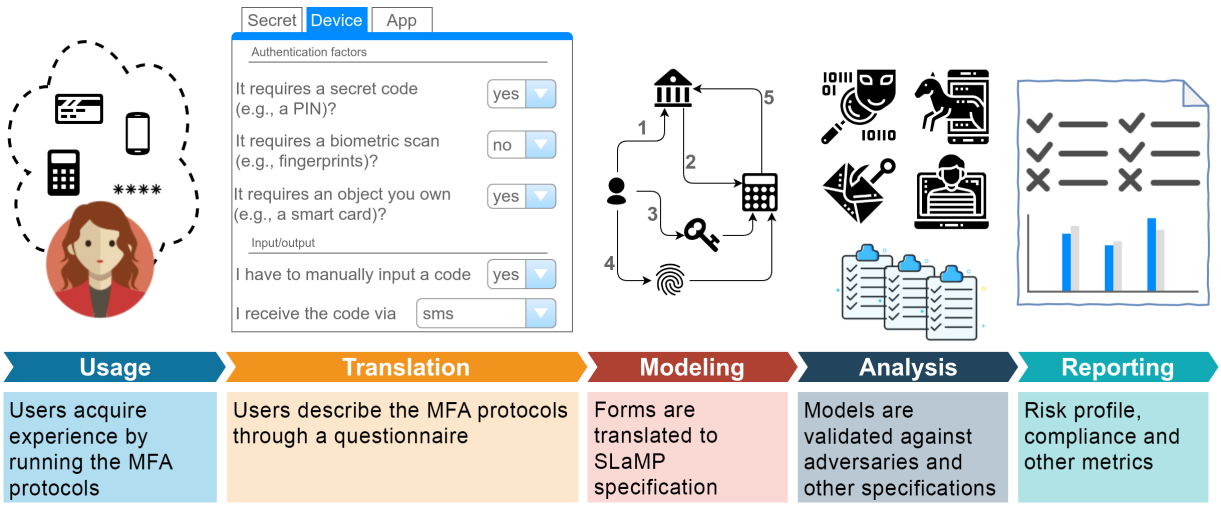


Figure 5.1: Flow of the proposed approach.

5.1 Goals

The primary goal of our tool is to allow, even non-expert users, for obtaining a security and usability assessment of a specified MFA protocol. In particular, it could help security experts to evaluate an MFA protocol at the very preliminary stages of the design process. Indeed, MuFASA could help to understand the consequences of certain design choices in terms of resistance against a set of attacker models.

On the other hand, MuFASA is built to help normal users to understand weaknesses and strengths of a protocol design and to raise their awareness of the risks they might be exposed to.

5.2 Implementation details

MuFASA has been developed as a Java application, based on the abstract architecture depicted in Figure 5.2. It consists of four main modules: the *Questionnaire* module, the *Translator* module, the *Analysis* module and the *Aggregator* module. Below we discuss them in detail.

5.2.1 Questionnaire

This module implements the user interface of MuFASA. After a first question regarding the Endpoint from which the protocol is started, the user is prompted with high-level questions about her experience with the MFA protocol she wants to model. Each round of questions aims

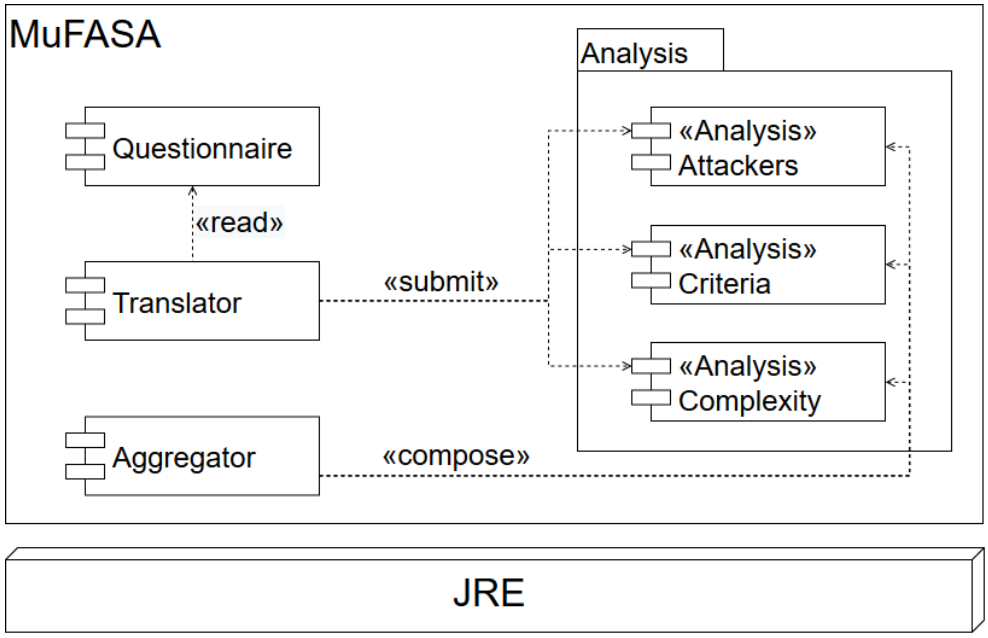


Figure 5.2: Architecture of MuFASA.

at precisely identify an authenticator. The rounds are iterated until the user describes all the authenticators in the protocol.

The actual questions presented at each round are determined by the previous answers. For instance, the first question is

"What is your *n*th operation?" (where *n* is the round counter)

The user can pick one of five answers, i.e., "I insert some secret credentials", "I read a value from a list", "I use a device", "I use a software" and "I send/receive something on my mobile phone (e.g., an SMS)". Some questions can be also accompanied by some pictures. The pictures show several elements and the user has to select the one that more resembles her experience. An example questionnaire is reported in the Appendix of this thesis.

5.2.2 Translator

The Translator module reads the answers to the questionnaire and converts them to a model of the MFA protocol, specified in SLaMP. The module follows an interpretation tree that exhaustively represents all the possible answers to a round of the questionnaire. Each leaf of the tree is labeled with the model of an authenticator. The list of all the possible sequences of forms is reported in the Appendix.

By combining all the authenticators in a sequence, the Translator obtains the protocol model. Eventually, the model is submitted to the analysis module.

5.2.3 Analysis

This module consists of a collection of sub-modules. Each sub-module implements a common interface: the *Analysis* interface. In this way, the set of analysis carried out by MuFASA can be extended by adding new sub-modules. For the time being, the three built-in analyses implement the operations described below.

5.2.3.1 Attackers

As anticipated in Section 4.1.1, each attacker corresponds to a function. Attackers' functions remove the authenticators and AFs from the target protocol until, eventually, it is entirely compromised. When an attacker cannot compromise a protocol, its function behaves as the identity. Moreover, applying the same attacker, i.e., the same function, twice on the same protocol has no effect. Thus, the module iterates until a fixed point is reached, i.e., the protocol is entirely compromised or all the attacks reduce to the identity. Eventually, the list of the attacks is returned. The list contains all the groups of attackers together with their effect on the protocol. Moreover, a list of unsuccessfully attackers is also provided.

5.2.3.2 Criteria

The security and best practices criteria evaluation amounts to a pattern matching between the protocol and the rules encoding each criterion. Each comparison results in a boolean value indicating whether the protocol matches the rule. The final result is a list of the criteria that the protocol matches.

5.2.3.3 Complexity

This module evaluates the complexity score of the specified MFA protocol. The complexity score is computed as the sum of three scores, i.e., memory, operations and devices (see Section 4.3).

5.2.4 Aggregator

The Aggregator module retrieves the output of each analysis module and combines them into a unified report. The report is then returned to the user in the form of a PDF document.

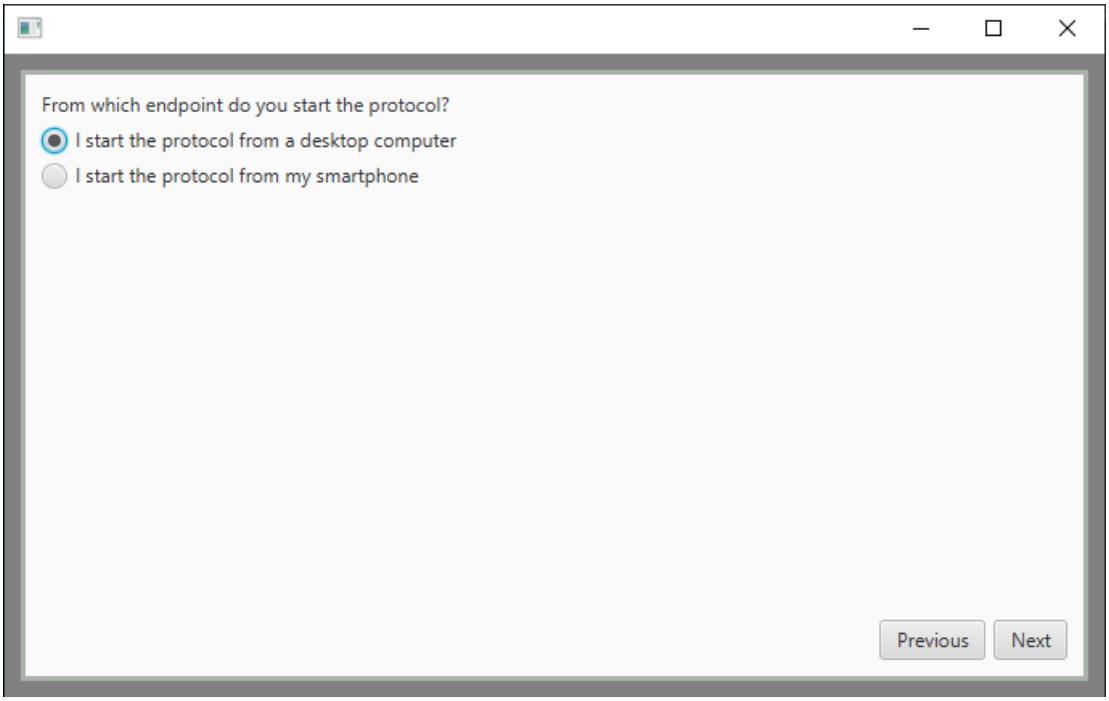


Figure 5.3: Screenshot of the first question proposed by MuFASA: the Endpoint choice.

5.3 Tool demo

MuFASA has been employed to perform part of a latitudinary study on MFA protocols adopted by 30 international banks. The results of the analysis, leveraging the data obtained with MuFASA, are presented in Section 6.2.

The tool is currently available (as an executable JAR) at `https://sites.google.com/fbk.eu/mufasa`. In the following, we present a demo, using Nordea1 as the protocol under analysis.

Demo. *Let us try to specify and analyze Nordea1 with MuFASA. Firstly, we run the jar and we select the wizard for protocol specification. A first question is asked, regarding the Endpoint from which the protocol will be executed (see Figure 5.3). We select the first option, since Nordea1 is supposed to be executed from a desktop computer. At this point, the wizard shows the page for specifying the authenticators composing the protocol (Figure 5.4). The upper part of the window shows the authenticators specified so far (none, for now). The lower part, instead, include the first part of the questionnaire for specifying various details of an authenticator. In our case, as our answer to the question "Which is your next operation?" we select the first option, i.e., "I insert some secret credentials" (modeling* 🔑 *in Nordea1). When pressing next, MuFASA will memorize the specified authenticator (no other questions are needed, for specifying the authenticator) and show again the page for specifiyng another authenticator (Figure 5.5).*

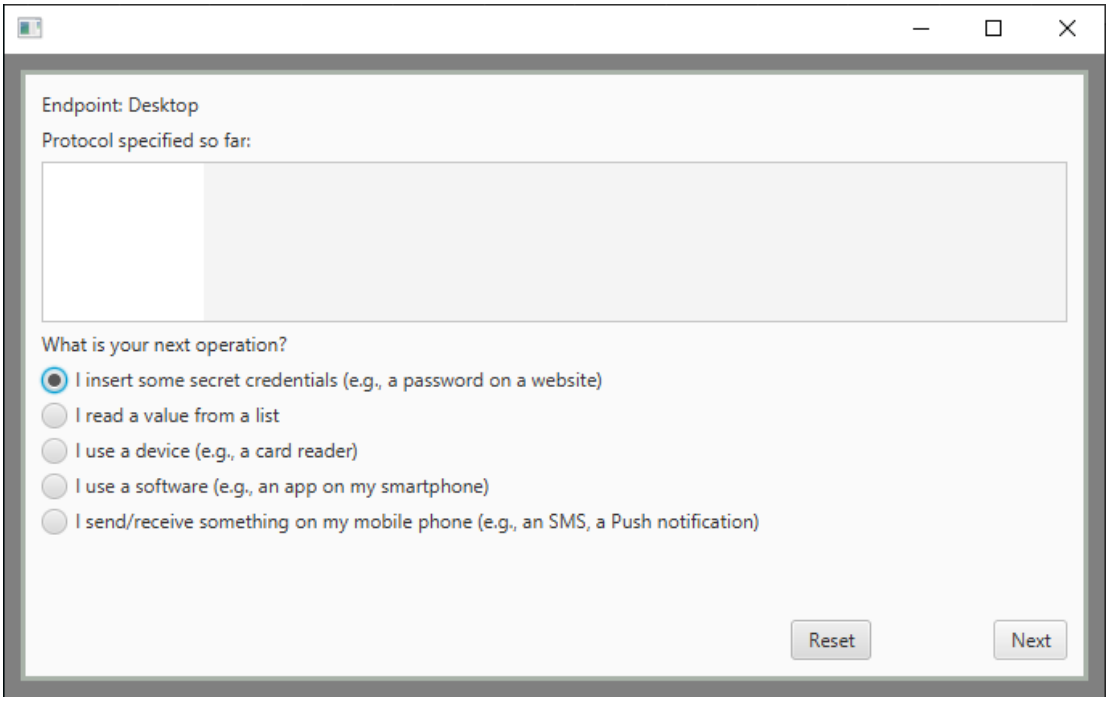


Figure 5.4: Screenshot of the wizard prompting for the specification of an authenticator.

This time, however, the page shows 🔑 *in its upper part, among the authenticators that we have specified already.*

The next authenticator to be specified is `opid` $\gg_h$ 🖩*[O,K]* $\gg_h$ `otp`*. After selecting the option "I use a device", new questions are shown. We have to indicate if it is a personal device (yes), if it is connected to something as an Endpoint (no), if it receives a sort of input code (yes, I personally digit it), if it shows information on the ongoing operation (no) and, finally, if the output has to be manually copied somewhere (yes). When the last answer is given, MuFASA saves the specified authenticator and redirects us again on the wizard page for specifying another authenticator (Figure 5.6).*

At this point, since the authenticator shown in the upper part of the page coincide with those of Nordea1, we can select the option "None, I'm authenticated" and proceed with the analysis.

Before starting the analysis, another wizard page is shown (see Figure 5.7). In this page, besides selecting the directory of the report file and its name, it is possible to define the set of attacker models to be considered in the analysis. By default, all the attacker models presented in Section 4.1.1 are considered. However, in the case that a user is interested in evaluating the protocol against a subset of them, she can easily include/exclude an attacker model by using the corresponding flag. Moreover, a user can select the maximum size of attacker combinations. In the case that a user wants to perform an analysis only on attacker models attacking singularly, this

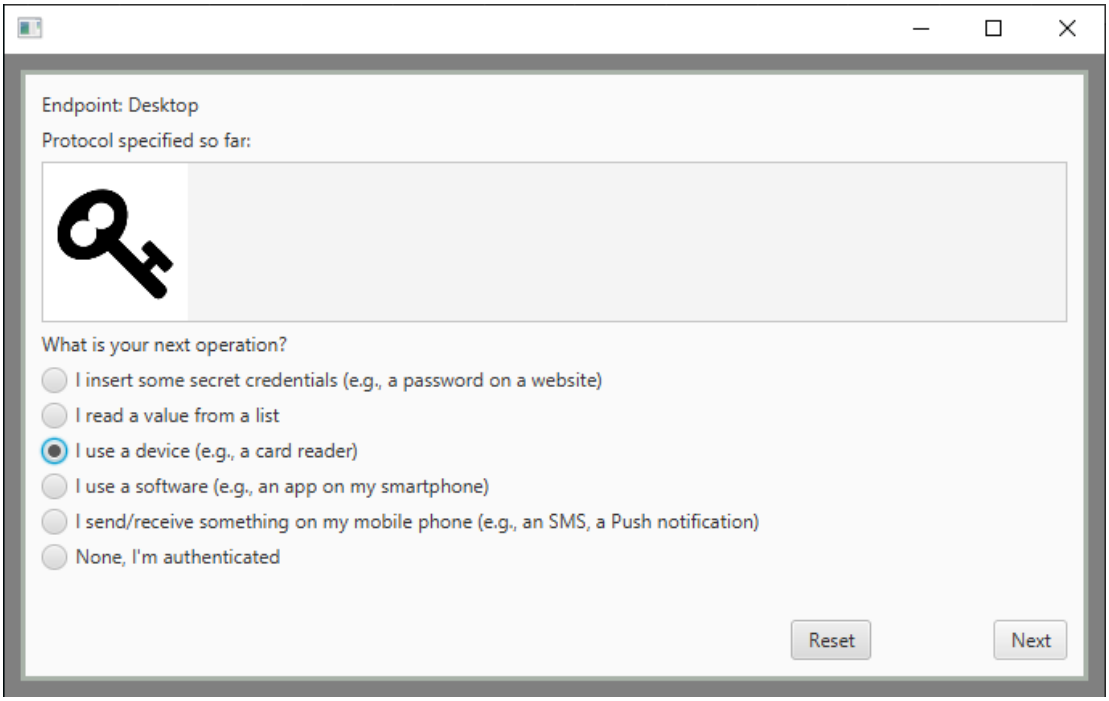


Figure 5.5: Screenshot of the wizard prompting for the specification of an authenticator, after specifying 🔑.

could be done by selecting the maximum size as 1. Finally, MuFASA allows to select the type of security analysis and report to be given as output. In particular, it offers the possibility to analyze a protocol and retrieve the minimum set of attackers compromising it (Minimum option). Alternatively, MuFASA can extensively analyze the resistance of a protocol against all the possible combination of attackers (Extensive) or considering all the attackers that compromise the protocol that are composed by sub-combination that do not (Selective).

It is worth noting that the evaluation of the protocol in terms of compliance with requirements (and best practices) cannot be customized. The same applies for the evaluation in terms of complexity.

As a final result, a PDF containing the outcomes of the three evaluations is reported.

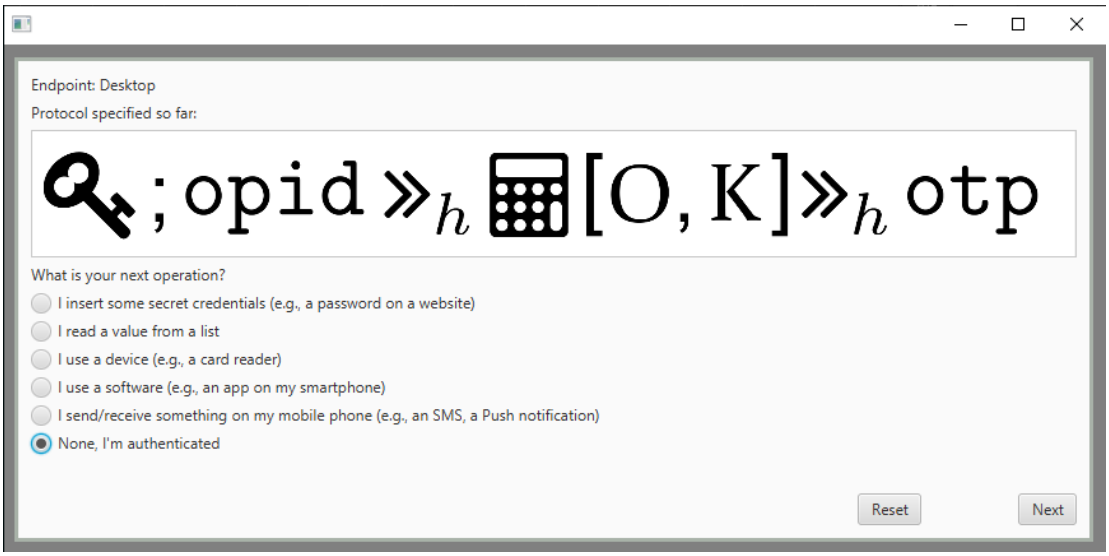


Figure 5.6: Screenshot of the wizard prompting for the specification of an authenticator, after specifying 🔑 and `opid` »h 🖩[O,K] »h `otp`.

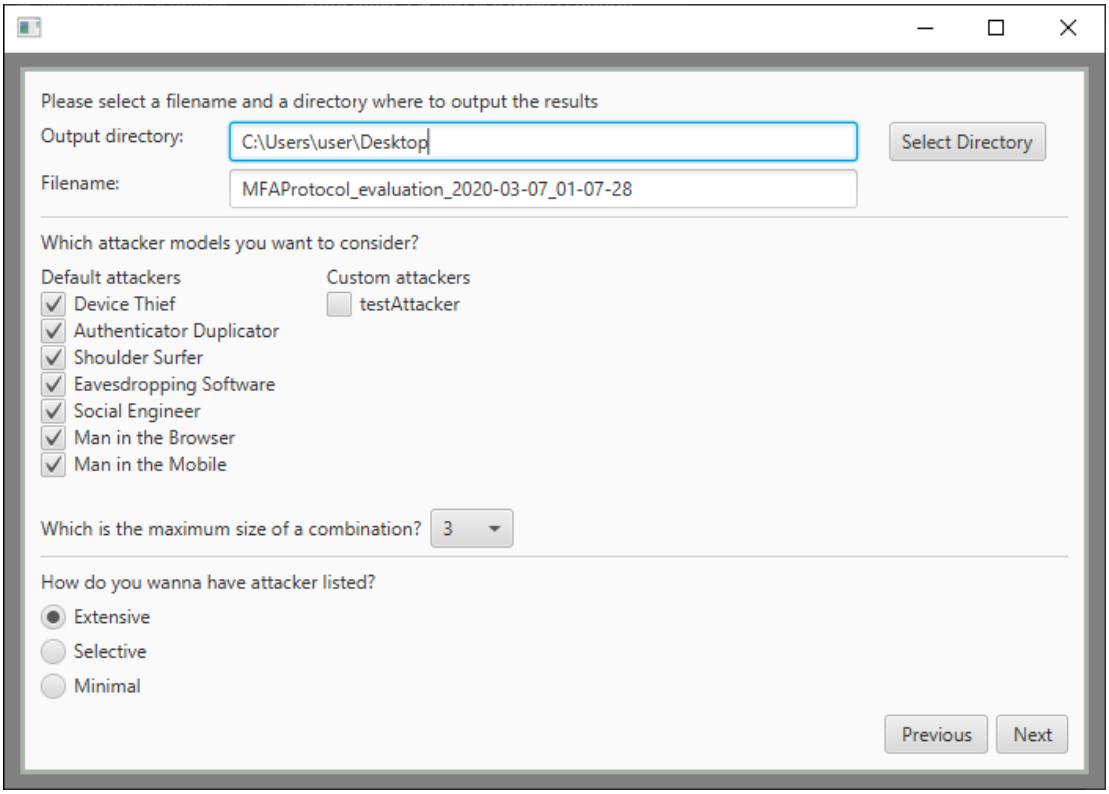


Figure 5.7: Screenshot of the wizard for customizing the security analysis and the output.

Chapter 6

Use Cases and Experimental Results

In this chapter we present two use cases on which our framework has been applied. In the eHealth use case, we applied the framework for designing an MFA solution for the eHealth services offered by Azienda Provinciale per i Servizi Sanitari (APSS), the local service provider for the Trentino province. The framework has been employed to support the protocol design phase, in order to evaluate the trade-off between complexity and security of the protocol.

In the online banking use case, we performed a latitudinary study on real MFA implementations adopted by 30 banks around the world. In this context, several aspects of each MFA implementation have been investigated, including the preliminary phases, the security level provided by the employed MFA protocols, their complexity, the compliance with security requirements and best practices.

6.1 eHealth Use Case

The eHealth online services manage Personal Health Records, medical data and, in general, very sensitive information that must be protected in a proper way. Indeed, a leakage or a theft of such data can cause serious damage to a user, with potentially deadly consequences.

This means that both security and privacy are of paramount importance in this scenario, which relevance is notably growing year by year (as reported in [Com18]).

However, along with the security and the privacy aspects, usability may take a relevant role in specific cases of eHealth services. Several medical conditions require constant monitoring, involving periodic transmission of (sensitive) data concerning one or more pathological conditions (e.g., glycemia levels for a person with diabetes). In such cases, the transmitted data have to be properly protected, while the employed security protocols must not discourage the user by

lacking in usability.

To sum up, eHealth services are a harsh application scenario for MFA, that must balance three different aspects in order to effectively protect users while not being felt as a burden in every day tasks.

The main eHealth services, at least in the European zone, are provided by government. During last years, the employment of MFA in eHealth services seemed to be an obvious yet necessary choice to guarantee a proper security level to protect even extremely sensitive set of data. Nevertheless, as reported in [CAM11], the majority of the European states are adopting ad-hoc security solutions, even leveraging different mechanisms, to protect citizen medical data.

In order to establish minimum security levels for handling and accessing medical data, the majority of national laws refer to the eIDAS directive [Eur14]. However, no detailed guidelines and design patterns are presented in such directive. Therefore, the currently adopted MFA solutions in the European eHealth context are quite various, providing different user experience and, most importantly, different security levels.

The chosen use case refers to a specific application of MFA in the eHealth services offered by Azienda Provinciale per i Servizi Sanitari (APSS), the local service provider for the Trentino province (in Italy). In particular, APSS needed to design a new MFA protocol for letting their user to access their Personal Health Records and telemedicine programs from their mobile devices (i.e., using an App as the Endpoint). The users would access their data via native smartphone apps, which can interact with other apps of the same developer. Moreover, users who adhere to telemedicine programs may require to access the applications four-six times a day. Therefore, the designed MFA protocol must not be difficult to execute. For the same reason, manual inputs for passing data between software authenticators and apps should be limited, and the users should not be forced to bring additional devices with them for the purposes of the authentication. Obviously, all the necessary preliminary phases for enrolling the users to the service and bind the authenticators to their identities have to be designed as well.

Besides realizing a new MFA protocol with limited availability of guidelines and design principles, other challenges have to be faced. Firstly, the stakeholders required the MFA protocol to employ - besides Memorized Secrets - only software authenticators, that could be easily installed on the smartphone from which a user may want to access her data. Secondly, the MFA protocol has to comply with security requirements and best practices in the MFA context. Finally, the MFA protocol must be easy to execute, since many users require to access their PHR multiple times a day for monitoring/inserting health values.

6.1.1 Framework application

Our methodology has been applied for the design of a new MFA protocol for accessing personal health records held by APSS (Azienda Proviciale dei Servizi Sanitari). Both the MFA protocol and the preliminary phases have been designed.

6.1.1.1 Design of the MFA Protocol

The designed MFA protocol had to guarantee both a low complexity and a good security level. Moreover, it had only to employ software authenticators. At the time, APSS had already developed a software authenticator, attesting both ownerhip and knowledge factors, for another MFA protocol. Therefore, such authenticator has been employed and adopted to the new MFA protocol (that had to be executed from the App).

The final design can be specified as: 🔑; 🔒[O,K] $\gg_i$ `otp`.

The security evaluation of such protocol highlighted the resistance against DT, AD, SS, ES (acting singularly). However, both SE and MM are successful.

For what concerns the compliance with security requirements, the protocol complies with **RL3** and **RL4**. Instead, it does not comply with **RL5** and **RL6**, since the generated `otp` is not bounded to a single operation and the user is not informed about the operation she is going to authorize. Moreover, the protocol complies with **BP2** (since the Android application has been developed using standard APIs) and **BP4**, since no SMS is used.

Finally, the complexity of the protocol is low: 2 (derived from the memorization of 🔑 and the PIN for unlocking the authenticator).

6.1.1.2 Design of the Preliminary Phases

The enrollment procedure for registering to the service has been designed to respect **RL7** and **BP5**. Therefore, given the availability of multiple offices and branches available on the territory, we decided to perform the identity proofing in person. This would indeed guarantee a high assurance level (as required by the eIDAS directive [Eur14]).

On the other hand, also the binding phase had to be defined. If possible, such phase should have respected **BP6**,**BP7**,**BP8**. To this aim, we decided that, at the time of the enrollment, the new user would receive a password (to be changed at her first access) and a "temporary access code", along with a part of an activation code to be inserted on the software authenticator (the other half will be sent via SMS on a phone number declared during the enrollment).

The designed Enrollment can hence be specified as 🏢. Furthermore, the two biding procedure

for the Memorized Secret and the Software authenticator can be specified as (**E**,–,🔒), (**E**,🔑,🔒). In the first case, the user requires the secret during the Enrollment, it does not require activation, and it is established by the user after inserting the temporary code. In the second case, the user requires the authenticator during its Enrollment, downloads the app and activates it providing the password and a composition of two activation codes (that will expire as soon as they have been successfully inserted in the authenticator).

In this case, the MFA implementation complies with **RL7**, **RL8** and **RL9**. Instead, for what concerns the best practices, it only complies with **BP5** and **BP6**, while does not respect **BP7** and **BP8** (since only the software authenticator is given).

6.1.1.3 Conclusions

The evaluation of the designed MFA implementation has been completed with the execution of a risk assessment procedure (following the OWASP Risk Management Framework [1]). The obtained results have been discussed with the stakeholders, that approved the proposed design. The described MFA implementation is currently employed for the APSS services.

6.2 Analysis of online banking services

Online Banking services are another scenario in which MFA has been widely adopted. These services allow customers to remotely access their bank accounts and financial data as well as to perform online payments and other financial transactions. These services are becoming increasingly popular among customers: according to Eurostat [Eur18], the number of European citizens using online banking services has doubled since 2007 and currently more than half of the European population use an online banking service daily.

The Online Banking services are one of the first scenarios in which MFA has been applied. Indeed, the European Banking Authority started to discuss about minimum security requirements and on the adoption of multi-factor authentication in 2011. In this scenario, the resources as the balance of an account, the possibility of transferring money or execute payment transactions, constitute sensitive data that must be properly protected against theft and other attacks. Therefore, during the years, MFA has been progressively adopted by the majority of banks. At the current time, an impressive number of MFA implementations are currently offered to customers of banks all around the globe.

However, the landscape of MFA implementations in Online Banking services is absolutely various. Indeed, the majority of banks offer to their customers a set of various MFA protocols,

[1] `https://owasp.org/www-project-risk-assessment-framework/`

differing in terms of resistance against threat models and user experience, but allowing the users to access the same sensitive data. On top of that, a relevant number of banks implement ad-hoc MFA protocols, which security level is uncertain and that may hide unknown vulnerabilities.

Important institutions and international authorities, as the European Banking Association, published some directives and regulations (as [Eur13a, Eur13b, Eur15, Eur17] to limit the variety of the available implementation and require a minimum level of security. Nevertheless, their influence on existing MFA implementations remains unclear.

Therefore, we decided to perform an analysis on the existing MFA solution adopted by banks operating in different countries, evaluating different aspects of the MFA implementations following different criteria and verifying the existence of possible correlations between them.

6.2.1 The Dataset

Therefore, we performed a latitudinal study on the adoption of MFA and the design choices made by 30 important international banks based in 10 different countries (3 banks per country). In particular, a first group includes 21 banks chosen among those based in the first seven countries in the European Union for gross domestic product[2], i.e., Germany, United Kingdom [3], France, Italy, Spain, Netherlands and Sweden. For each country, we selected the three largest banks (offering online banking services to individuals) in terms of assets (according to the Standard & Poor's 2017 classification[4]). solutions adopted by 30 important international banks based in 10 different countries (3 banks per country).

The second group of banks is taken from relevant countries (for the banking sector) that are not subject to the EU legal framework, i.e., China, USA and Switzerland. Again, for these countries we considered the three largest banks in terms of assets. The geographic distribution of the selected banks is shown in Figure 6.1. This allows us to enlarge our perspective on the banking sector by investigating how non-EU banks behave with respect to MFA adoption.

For each of the considered banks, we considered the MFA protocols offered to the customers, distinguishing from those for executing payments from a Desktop computer (Internet Payments, IPs from now on) and those from smartphone (Mobile Payments, MPs from now on). Obviously, we considered every aspect concerning the implemented MFA protocols. Besides the MFA protocol themselves, for each bank we considered the process of identity proofing, all the authenticators that are offered to the customers and their binding phases. Furthermore, the adoption of the so-called exemptions (i.e., the possibility to avoid the execution of an MFA protocol in low risk situations) has been investigated.

[2] https://goo.gl/ZctkLc

[3] The data collection started before Brexit.

[4] https://goo.gl/HR3HDQ

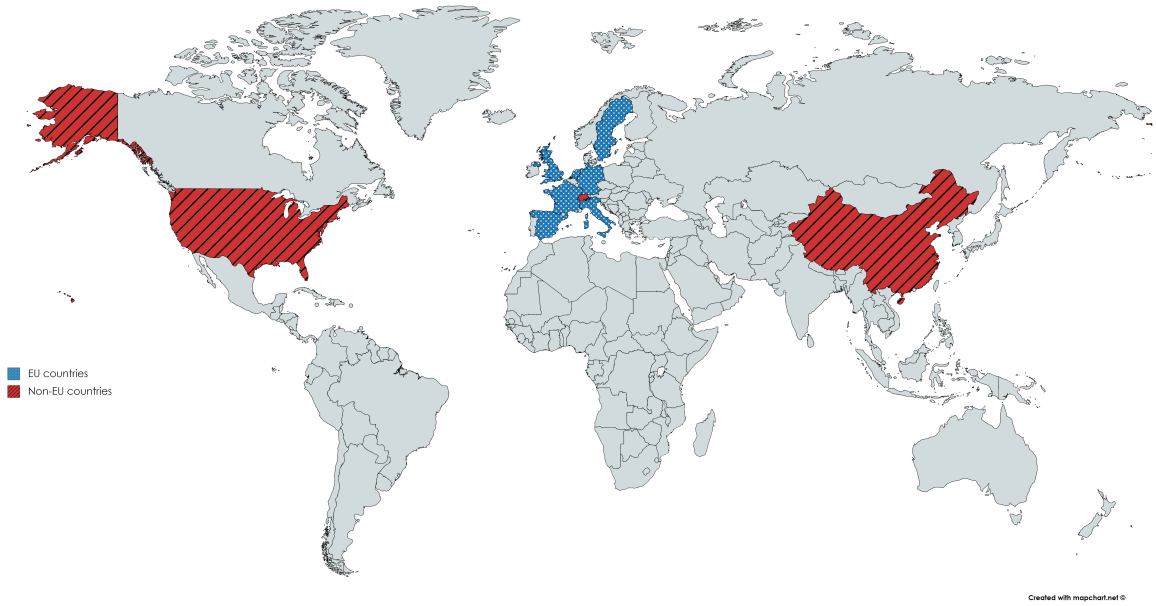


Figure 6.1: Geographic distribution of the analyzed banks. EU countries (filled with a blue pattern) are Spain, France, UK, Netherlands, Italy, Germany and Sweden. Non-EU countries (filled with a red pattern) are USA, Switzerland and China.

6.2.2 Evaluation Criteria

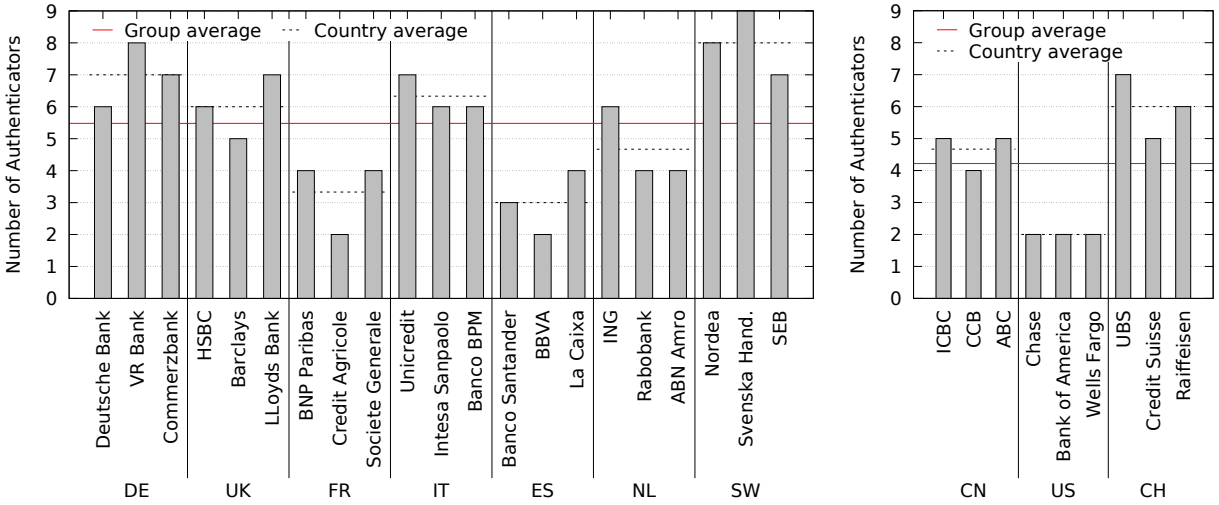
The selected banks have been evaluated according different aspects.

Firstly, banks have been evaluated in terms of compliance with security requirements (see Section 4.2.1) and best practices (see Section 4.2.2), following the specific criteria presented in the respective sections.

Then, the considered banks are evaluated in terms of resistance against threats (see Section 4.1.1) and complexity (see Section 4.3). To this aim, each MFA protocol employed by the considered banks is evaluated in terms of resistance against the set of attackers presented in Section 4.1.1. Instead, to evaluate a bank in terms of complexity, we compute the average complexity scores of all MFA protocols that the bank offers to its customers.

6.2.3 Results of analysis

In this section, we present the results of our investigation. In particular, we firstly present our findings in terms of demographic aspects of the considered dataset (see Section 6.2.1). Then, we present our evaluation by means of the data and criteria discussed Section 6.2.2. Moreover, we



(a) EU banks (b) non-EU banks

Figure 6.2: Number of authenticators provided by banks.

verify whether some correlations between the obtained results exist.

6.2.3.1 Demographics

To characterize the landscape of how MFA has been adopted in the online banking sector, we consider several perspectives. Below we present some statistics on authenticators, MFA protocols, exemptions, and enrollment and binding procedures.

6.2.3.1.1 Authenticators. Figure 6.2 shows the number of authenticators that each bank offers to its customers. Vertical bars indicate the number of distinct authenticators per bank; horizontal lines indicate the average per nation (blue dashed line) and per group (red solid line). All banks employ a minimum of 2 and a maximum of 9 authenticators. At national level, values appear homogeneous. As a matter of fact, the maximum variation in the number of authenticators per nation is 2 and the average variation per nation is 1.7. This may indicate that some national trends exist. These trends may be the result of national laws, market strategies or adoption of national identity systems (e.g., BankID [Ban] for Swedish banks).

Although the number of employed authenticators provides an indication on the variability of the authenticators commonly offered by banks, we are also interested in their type (see Section 6.2.1). Figure 6.3 shows the distribution of authenticators per type, i.e., devices, software, look-up secrets and memorized secrets, adopted by EU and non-EU banks. White bars indicate the percentage of banks that employ at least one authenticator of the given type. Moreover, for de-

vice and software authenticators, the figure shows the percentages with respect to sub-categories, i.e., single-factor, multi-factor and out-of-band. Note that, in order to differentiate the employment of credentials (username and password) and additional memorized secrets, we represent them in two separate columns, namely "credentials" and "2nd memorized secret", respectively.

We observe some facts. All considered banks provide their users with credentials. Secondly, almost all of them (except one) offer at least one device authenticator. Moreover, EU banks offer 1.4 device authenticators on average, while the average number of device authenticators for non-EU banks is 1.9. Among device authenticators, out-of-band authenticators (i.e., SIM cards) are more common (48% for EU and 78% for non-EU banks). We can also observe that EU banks employ multi-factor device authenticators more frequently than single-factor device authenticators (38% and 29% of the banks, respectively), whereas non-EU banks do the opposite (with 67% and 33% of non-EU banks employing single and multi-factor device authenticators, respectively).

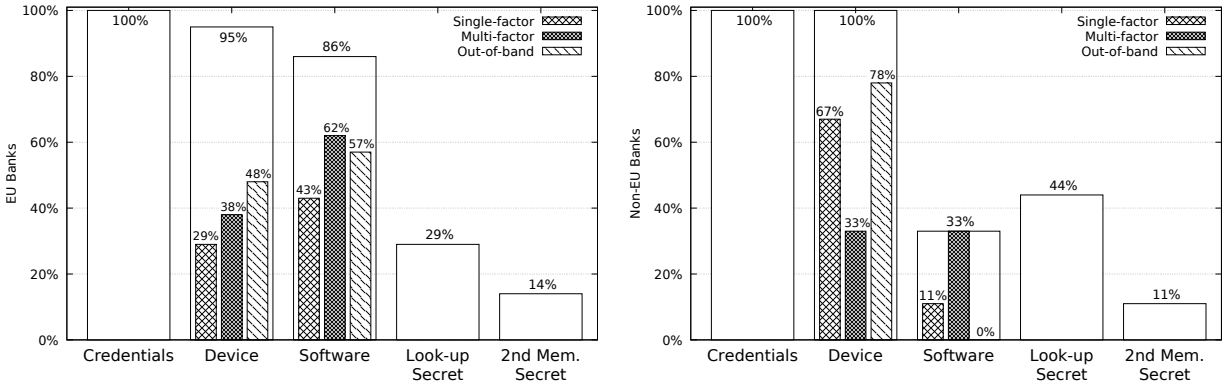
The adoption of other types of authenticators differs between the two bank groups. For EU banks, the second most frequent type of employed authenticators is software. As a matter of fact, 86% of EU banks adopt at least one authenticator of this type.[5] Among these banks, the average number of software authenticators is 3.1. Multi-factor ones are dominant (62%), although a significant number of out-of-band software authenticators are also employed (57%). Single-factor authenticators are less common (43%). Look-up secrets and extra memorized secrets follow in the order, being employed by 29% and 14% of EU banks, respectively.

On the other hand, non-EU banks present a different trend. After device authenticators, look-up secret is the second most adopted type of authenticator (44%). Software authenticators are employed by 33% of the banks. All banks adopting software authenticators provide at least one that is multi-factor. Only one non-EU bank provides also a single-factor software authenticator. Finally, only 11% of the banks in this group adopt additional memorized secrets.

6.2.3.1.2 MFA protocols. Our analysis reveals that banks usually provide their clients with a variety of MFA protocols. For each bank, we distinguish MFA protocols for IP and for MP. These two kinds of payment methods differ for the endpoint on which they are executed. This distinction is necessary to evaluate the compliance with requirements and best practices as well as to analyze the security and complexity of MFA protocols (see following sections). Figure 6.4 shows the number of MFA protocols adopted by each bank. The figure also reports the average number of MFA protocols for IP (red lines) and MP (blue lines), for each nation (dashed lines) and for bank group (solid line). Overall, we counted 32 distinct MFA protocols employed by banks for IP and 29 protocols for MP.

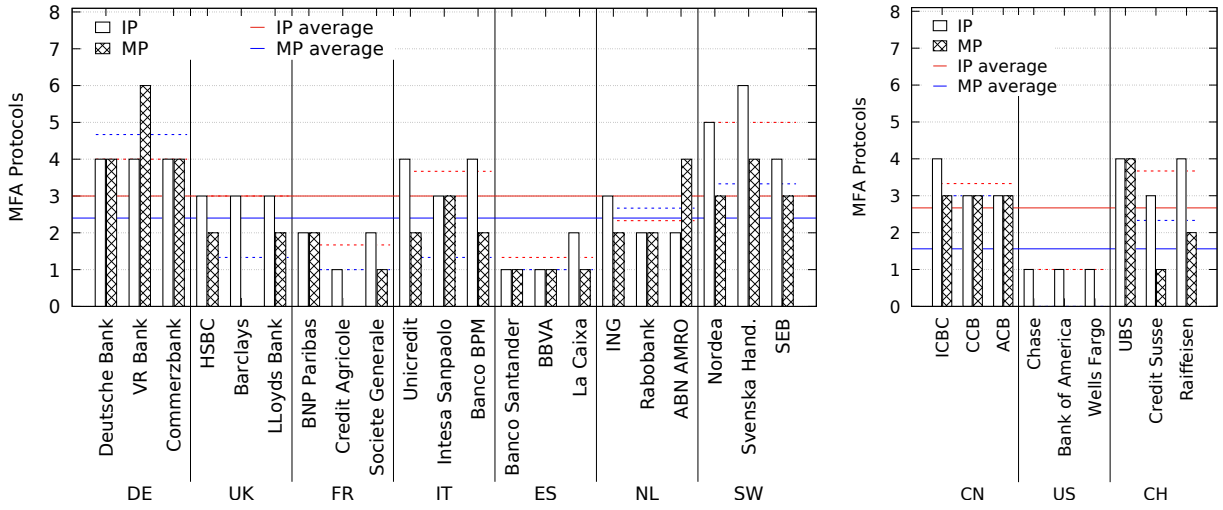
We observe a few facts. Except for two banks, the number of MFA protocols for IP is equal

[5]Note that every bank has an official mobile application, but here we only consider those that act as an authenticator.



(a) EU banks (b) non-EU banks

Figure 6.3: Percentage of authenticators types offered by banks.

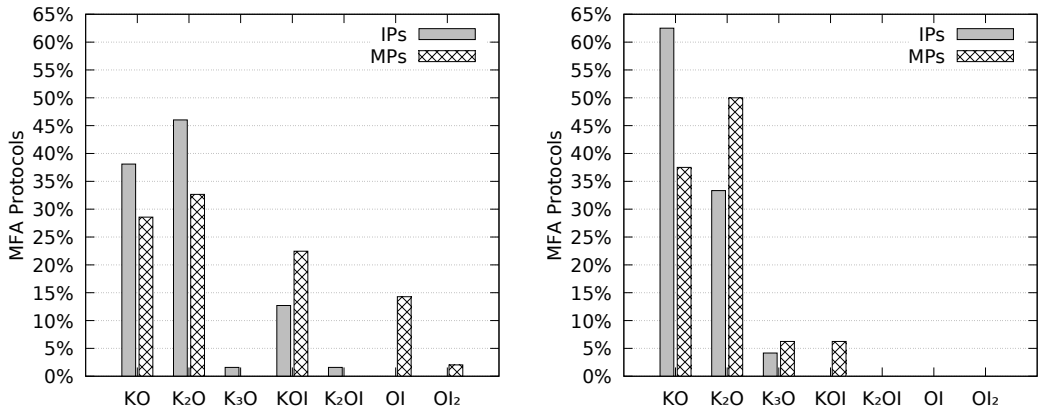


(a) EU banks (b) non-EU banks

Figure 6.4: Number of MFA protocols supported by banks.

or greater than the one for MP. In particular, four banks do not support MFA protocols for MP at all.[6] As expected, we observe a correspondence between the number of authenticators (Figure 6.2) and the number of MFA protocols per bank. As a matter of fact, most authenticators are associated to a limited number of MFA protocols (often only one). Moreover, most of the MFA protocols (around 80%) rely on two authenticators.

[6]Note that the absence of MFA protocols does not imply that MP is not supported. In fact, Barclays, Credit Agricole and all American banks do support MP – even though only low risk operations are supported.



(a) EU banks

(b) Non-EU banks

Figure 6.5: Combinations of AFs used in MFA protocols.

To better understand the differences between the adopted MFA protocols, we investigated how many (and which kind of) AFs are used by them. Figure 6.5 shows the result of this analysis, displaying the combinations of AFs that the analyzed MFA protocols employ. Each combination can be composed by **K**nowledge, **O**wnership and **I**nherence factors. If more than one factor of the same type is used, the number of factors is reported as a subscript for the corresponding type. For example, if an MFA protocol leverages a memorized secret and a multi-factor authenticator device attesting both ownership and knowledge factors, such an MFA protocol is annotated with combination K_2O.

Figure 6.5 shows that a large number of MFA protocols leverage more than two AFs (the minimum for an MFA protocol) for authenticating the user. This fact is particularly noticeable for EU banks, where 52% of the employed MFA protocols for IP and 59% of those for MP leverage at least three AFs. A slightly different situation can be observed for non-EU banks. In this group, only 37% of the MFA protocols for IP rely on more than two AFs. In the case of MP, however, the trend is opposite, with 62% of the MFA protocols employing at least three AFs.

We also observe that, both for IP and MP, combinations involving knowledge and ownership factors are the most frequent. The employment of inherence factors is more frequent in MFA protocols for MP employed by EU banks. In particular, 38% of these protocols employ at least one inherence factor and around 15% of them leverage only inherence and ownership factors. Interestingly, this type of AF is usually not employed by non-EU banks, where only 6% of the MFA protocols adopted by those banks leverage an inherence factor. Finally, no combinations constituted only by knowledge and inherence factors have been observed.

Table 6.1: Exemptions, Enrollment and Binding procedures per bank.

	Deutsche bank	VR Bank	Commerzbank	HSBC	Barclays	Lloyds bank	BNP Paribas	Credit Agricole	Societè Générale	Unicredit	Intesa Sanpaolo	Banco BPM	Banco Santander	BBVA	La Caixa	ING	Rabobank	ABN AMRO	Nordea	Svenska Handelsb.	SEB	ICBC	CCB	ABC	Chase	Bank Of America	Wells Fargo	UBS	Credit Suisse	Raiffeisen
Country	DE			UK			FR			IT			ES			NL			SW			CN			US			CH		
Exemptions	✔	✔	✔	✔✔	✔✔	✔✔	✔✔	✔✔	✔	✔✔	✔✔	✔	✔✔	✔	✔✔	✔	✔✔	✔✔	✔✔	✖	✖	✔	✔	✔	✔✔	✔✔	✔✔	✔✔	✔✔	✖
Enrollment	🏢	🏢	🌐	🏢	🏢	🏢	🌐	🌐	🌐	🏢	🏢	🏢	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🏢	🌐	🏢	🏢	🏢	🌐	🌐	🌐	🌐	🌐	🌐
Binding Request	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🏢	🏢	🏢	🌐	🌐	🌐	🌐	🌐	🌐
Delivery	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🌐	🏢	🏢	🏢	🌐	🌐	🌐	🌐	🌐	🌐
Activation	🌐*	🌐	🌐*	🔒	🔒	🌐	🌐*	🌐	🌐*	🌐	🔒	🔒	🌐	🌐	🔒	🌐	🌐	🌐	🌐*	🔒	🌐	🏢	🏢	🏢	🌐	🌐	🌐	🔒	🌐	🌐*

6.2.3.1.3 Exemptions The adoption of exemptions can influence both the security level and perceived ease-of-use of MFA protocols. To this end, we investigated the type of exemptions allowed by each bank. The consequent evaluation in terms of ease-of-use (**BP3**) will be discussed in the following sections.

The type of exemption for every bank is reported in Table 6.1. We can observe from the table that exemptions are widely adopted. As a matter of fact, 27 of the considered banks adopt some form of exemption. The adopted level of exemptions seems to be homogeneous for each country. The only three banks that do not support exemptions are located in two countries, i.e., Sweden and Switzerland.

6.2.3.1.4 Enrollment The enrollment phase plays a critical role in MFA. The analysis of the offered modalities allows to understand at which level of security the verification of user identity is performed. This information will be used in the next section to assess the compliance of banks with **RL7**, **RL8** and **RL9**.

The enrollment modalities offered by banks are reported in Table 6.1. We observed that every bank allows enrollment at the bank. In the table, symbol 🌐 indicates the possibility for the user to choose between remote or *de visu* enrollment, whereas symbol 🏢 indicates that only the second option is available for a given bank. We can observe that remote enrollment is fairly common. Indeed, out of 30 banks, 18 allow remote enrollment. Also in this case, values appear homogeneous at a national level.

6.2.3.1.5 Binding The binding of an authenticator to a user's identity can influence the security of MFA protocols leveraging that authenticator. Here, we analyze the modalities offered by banks for binding, which will allow us to evaluate the compliance of banks with requirements **RL8** and **RL9**.

Table 6.1 presents the worst case (in terms of security) of the binding procedures offered by every bank. Further details on the enrollment and binding procedures adopted by all banks are given in the supplementary material. Intuitively, the analysis of the worst case provides an indicator of the compliance of banks with **RL8** and **RL9** and, in particular, the resistance of a procedure to attacks: if even in the worst case an attacker is not able to compromise an authenticator, the others will be reasonably secure.

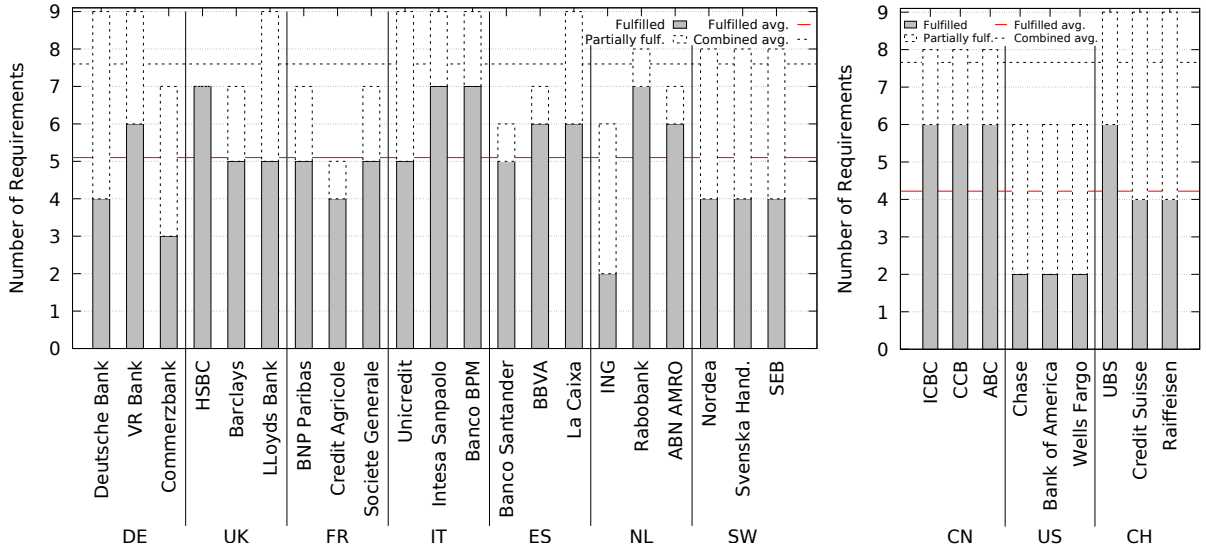
From the table, we observe that the remote request of authenticators is massively supported. A similar trend can also be observed for the delivery of authenticators. The only exception is represented by Chinese banks in which all binding operations – request, delivery and activation – have to be performed at the bank. The majority of banks also allow a remote activation of authenticators, but using a weak procedure. Only 12 banks (among the 30 considered) ensure an adequate level of security for the activation of authenticators (either requiring clients to activate them at the bank or though MFA leveraging previously activated authenticators). Six of these banks would actually offer an activation step leveraging an MFA protocol, but the employed authenticators have not been bound with a sufficient security level. In Table 6.1, this is marked with 🌍*. An example is the binding procedures offered by Commerzbank and BNP Paribas: those banks offer the possibility to activate a software authenticator through an MFA protocol based on the reception of an SMS on an out-of-band device. However, the binding of the out-of-band device can be performed remotely, hence not complying with **RL8** and lowering the security level of the binding procedures relying on it.

6.2.3.2 Compliance with requirements and best practices

In this section we discuss the compliance of banks with the requirements and best practices presented in Section 6.2.2. A complete view of the analysis is reported in the supplementary material.

6.2.3.2.1 Requirements Figure 6.6 shows, for each bank, the number of fulfilled (solid bar) and partially fulfilled (dashed bar) requirements. The average number of fulfilled requirements is represented by a solid (red) line and the average number of fulfilled and partially fulfilled requirements by dashed (blue) line.

We observe that none of the considered banks does meet all nine identified requirements. For EU banks, the average number of requirements fulfilled by banks is 5.1. If we combine both fulfilled and partially fulfilled requirements, the average increases to 7.6. The number of fulfilled requirements appears to be homogeneous among the banks of the same country (average variation is 2). The maximum variation is more than 2 only for Dutch and German banks (5 and 3, respectively). However, we observed some differences in the level of compliance for different countries. Three countries – Germany, France and Sweden – have an average of fulfillment below 4.6. On the



(a) EU banks

(b) non-EU banks

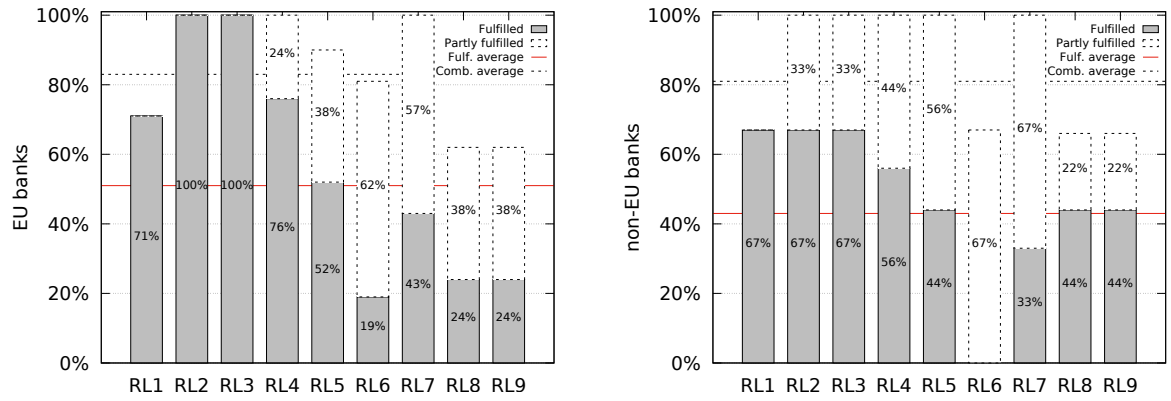
Figure 6.6: Fully and partial fulfilled requirements per bank.

contrary, Italian banks adhere to regulations and directives more than others, with an average of 6.3 fulfilled requirements.

For non-EU banks, the average number of fulfilled requirements is 4.2. This is not surprising, since our survey focused on requirements derived from EU regulations and directives. However, a deep look at the data shows that this value is strongly influenced by the low number of requirements met by US banks, which fulfill only two requirements. On the contrary, all Chinese banks comply with six requirements, which is higher than the average number of requirements met by EU banks. This means that, even if they are not subject to the same regulations and directives as EU banks, Chinese banks are aligned to EU security requirements. Similar observations apply to Swiss banks. Even if the average number of requirements met by these banks is not as high as the one met by Chinese banks, it matches the average number (5.1) met by EU banks.

We now analyze the compliance of banks with single requirements. Figure 6.7 shows the percentage of banks that fulfill each requirement. For each requirement, the gray bar indicates the percentage of banks that fulfill the requirement and the dashed bar the percentage of banks that partially fulfill it.

RL1, which concerns integrity checks on multi-purpose devices, is met by 71% of EU banks. In particular, none of the Swedish banks meet this requirement, along with two Dutch and one French bank. However, it is worth noting that this requirement will enter into force in the first half of 2019 [Eur13a] and, thus, EU banks do not to have to comply with it yet. For what concerns non-EU banks, 67% of them comply with **RL1**. Interestingly, all US banks comply with this



(a) EU banks

(b) Non-EU banks

Figure 6.7: Percentage of banks that fully and partially comply with the requirements.

requirement.

RL2, which requires the employment of MFA for risky operations, is fulfilled by all EU banks. This meets our expectations, since a first definition of this requirement [Eur13a] was introduced in 2014. This requirements is also met by all Chinese and Swiss banks, but by none of the US banks.

RL3 and **RL4** concern the usage of distinct and independent authentication factors in MFA protocols, respectively. While the first one is fulfilled by all EU banks, the second is only met by the 76% of them. The remaining 24% of EU banks, however, partially fulfill **RL4**, since they offer at least one MFA protocol employing two authenticators. This is due to the fact that some EU banks employ MFA protocols that only leverage a mobile application attesting both inherence and ownership factors (hence not leveraging independent AFs). Similarly to **RL1**, both these requirements were introduced in [Eur13a], which will enter into force in 2019. However, the high level of compliance with **RL3** and **RL4** might indicate that EU banks have already taken actions to adhere to this regulation. The percentage of non-EU banks that comply with **RL3** and **RL4** is 67% and 56%, respectively.

RL5, which requires the generation of a unique authentication code in every MFA execution, is fulfilled by 52% of EU banks and partially fulfilled by 38% of them. This can be explained by the fact that half of the banks offer a large range of heterogeneous MFA protocols where at least one does not employ an `otp` generated using an `opid`. It is worth mentioning that the fulfillment of this requirement will become mandatory only in 2019. For non-EU banks, instead, we have 44% of them complying with **RL5**, while the others (56%) partially fulfill it.

The level of compliance with **RL6** is the lowest, when compared to those of the other requirements. Indeed, this requirements is met by 19% of EU banks and partially fulfilled by 62% of

them. In words, this means that the majority of the banks (81%) employ at least one MFA protocol that does not inform the user about what operation she is authorizing. Similarly to **RL5**, the compliance with this requirement will become mandatory in 2019. It is worth noting that none of the non-EU banks comply with **RL6**, while 67% of them partially fulfill it.

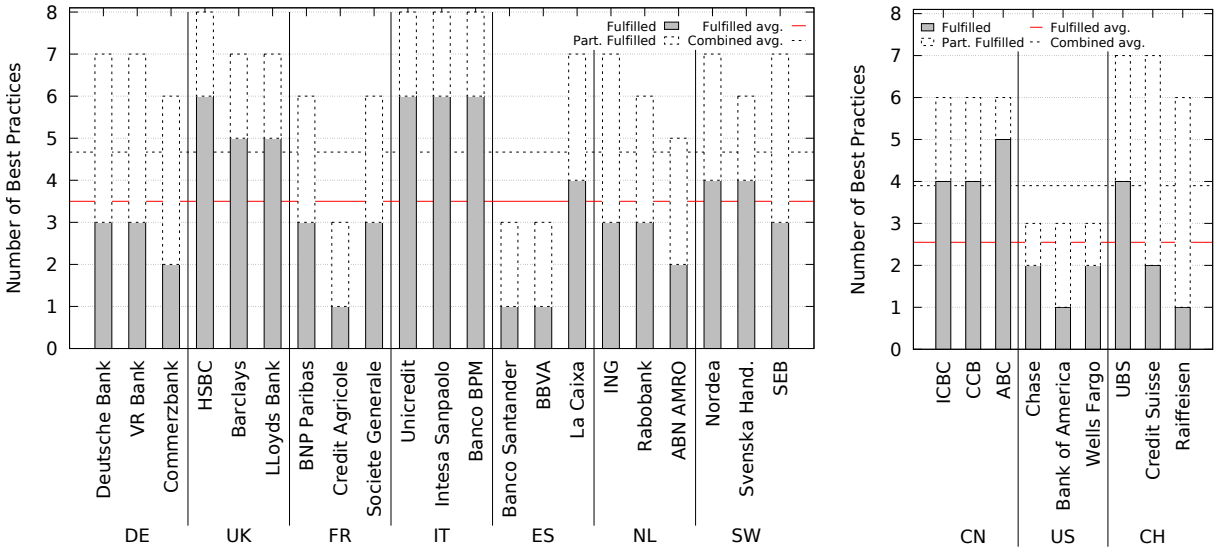
The fulfillment of **RL7**, which concerns user enrollment, strongly reflects the results presented in Section 6.2.3.1. Being influenced by enrollment modalities, **RL7** is fulfilled by those banks only providing enrollment in their branches (43% and 33% of EU and non-EU banks, respectively). The other banks fulfill **RL7** only partially.

Finally, we analyze requirements **RL8** and **RL9**. Recall from Section 6.2.2 that **RL8** concerns the level of security of the binding phases, whereas **RL9** concerns the activation of remotely delivered authenticators. These requirements exhibit the same level of compliance: 24% of EU banks fulfill these requirements. In several cases, banks employ authentication protocols leveraging multiple factors for activating an authenticator; however, the binding of these authenticators is performed without a proper security level, causing the requirements not to be fulfilled. Similarly to other requirements, **RL8** and **RL9** will enter into force in 2019. Among non-EU banks, 44% of them comply with both **RL8** and **RL9**, whereas 22% partially fulfill them.

To summarize, EU banks comply, on average, with half of the considered requirements. This may be due to the fact that five of them are specified on directives (and regulatory standards) entering into force only in 2019. Indeed, the percentage of EU banks complying with **RL1**, **RL5**, **RL8**, **RL9** and especially with **RL6** is low. However, a large number of EU banks offers at least one MFA protocol that meets all criteria defined in Table 4.14, thus partially fulfilling these requirements. Therefore, the majority of the banks can easily become compliant with these requirements just by offering a subset of the MFA protocols they currently support. When considering both the fulfillment and partial fulfillment of **RL5** and **RL6**, more than two-third of EU-banks comply with these requirements. If we assume that all requirements that are currently partially fulfilled will be fully met by 2019, EU banks will comply, on average, with more than 7 requirements out of 9.

6.2.3.2.2 Best practices We also analyzed the compliance of banks with best practices. Figure 6.8 shows the result of our analysis. For each bank, the number of fulfilled best practices is represented by a solid bar and the number of partially fulfilled best practices by a dashed bar. In the figure, we also report the average number of fulfilled best practices (solid line) and the average number of fulfilled and partially fulfilled best practices (dashed line).

As for the requirements, no bank fulfills all eight identified best practices. The average number of best practices fulfilled by a bank is 3.5 and 2.5 for EU and non-EU banks, respectively. If we consider both fulfilled and partially fulfilled best practices, the average number is 4.6 and 3.9, respectively.



(a) EU banks

(b) non-EU banks

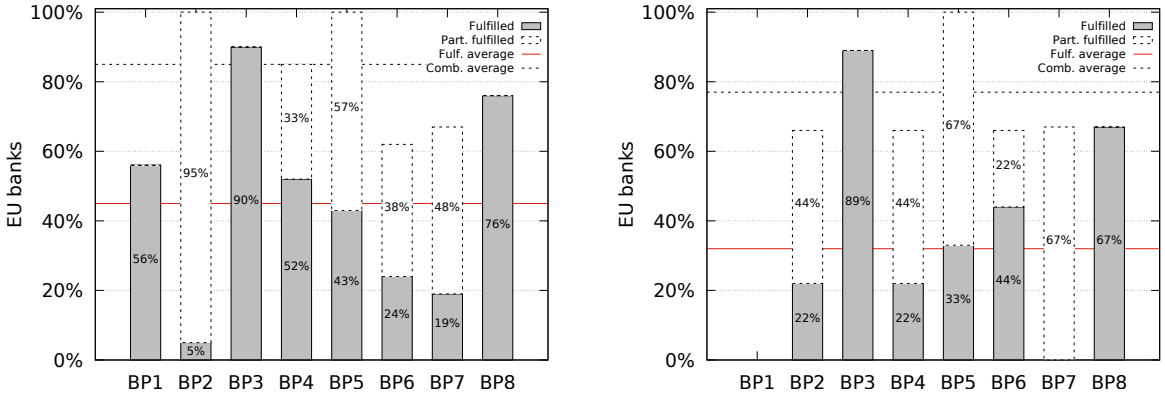
Figure 6.8: Fully and partial fulfilled best practices per bank.

Among EU banks, Spain is the country in which banks meet less best practices, with an average of two best practices. On the other hand, Italian banks lead also in terms of fulfilled best practices, with an average of 6 best practices. Among non-EU countries, the results for best practices are similar to the ones concerning requirements. US banks fulfill an average of one best practice, whereas Chinese banks an average of 4.3 best practices. The position of Swiss banks is quite heterogeneous with a variation in the number of fulfilled best practices equal to 3 (the average number of fulfilled best practices is 2.3).

To gain more insights, we analyzed the compliance of banks per best practice. Figure 6.9 shows the percentages of banks that fulfill and partially fulfill each best practice. From the figure, we observe that the percentage of banks that met the best practices is lower than the one of banks that met the requirements (45% compared to 51% for EU banks and 32% compared to 43% for non-EU banks). However, if we consider both the fulfillment and partial fulfillment of requirements and best practices, the percentage of banks are aligned (85% vs. 83% and 77% vs. 81%, for EU and non-EU banks, respectively).

We now discuss the compliance with each best practice against the fulfillment criteria in Section 6.2.2.[7] **BP1**, which concerns the integration of software authenticators in mobile banking applications, is fulfilled by 56% of EU banks. On the other hand, 67% of non-EU banks fulfill this best practice because they do not offer any software authenticator. The software authenti-

[7]Recall that **BP5** and **BP6** are subsumed by **RL7** and **RL8**. In the following, the same observations apply to both the requirements and the best practices.



(a) EU banks

(b) Non-EU banks

Figure 6.9: Percentage of banks that fully and partially comply with the best practices.

cators offered by the remaining 33% (i.e., the Swiss banks) are instead not integrated with the respective mobile banking applications, thus violating **BP1**.

BP2 concerns the usage of commonly used libraries to execute security-relevant operations. The fulfillment of this best practice is 5% for EU and 22% for non-EU banks. It is worth noting that all Chinese banks do not comply with this best practice. This may be due to the fact that the majority of services on which Android applications rely on (e.g., Google Play Services, Firebase) are blocked by the Chinese firewall [CNB, Goo17]. The least used APIs are those concerning the integrity checks and keystore, while those related to SSL are always implemented. Moreover, we observed that several software authenticators use commercial solutions instead of those we considered in Section 6.2.2. We will discuss this aspect in Section 6.3.

BP3, which concerns the adoption of step-up authentication mechanisms, is fulfilled by 90% and 89% of EU and non-EU banks. Our analysis revealed that almost all banks employ some form of exemption (see Table 6.1), thus supporting step-up authentication.

BP4 concerns the usage of SMS messages in MFA protocols. 52% of EU banks do not employ any MFA protocol leveraging SMS messages, whereas 15% of the same group of banks employs only MFA protocols relying on them. For what concerns non-EU banks, only 22% of them employ MFA protocols that do not use SMS messages while 33% of them (specifically, US banks) employ only MFA protocols relying on SMS.

BP7 concerns the binding of two physical authenticators immediately after the enrollment. This best practice is fulfilled by 14% of EU banks and never fulfilled by non-EU banks. However, 48% and 67% of EU and non-EU banks (respectively) partially fulfill **BP7**. It is also worth noting that 33% of both EU and non-EU bank violates the best practice, not offering the user any physical authenticator immediately after her enrollment. Finally, **BP8** concerns the availability

of multiple types of authenticators. This best practice is fulfilled by 76% of EU banks and 67% of non-EU banks. It is worth noting that all non-EU banks that violate this best practice are US banks.

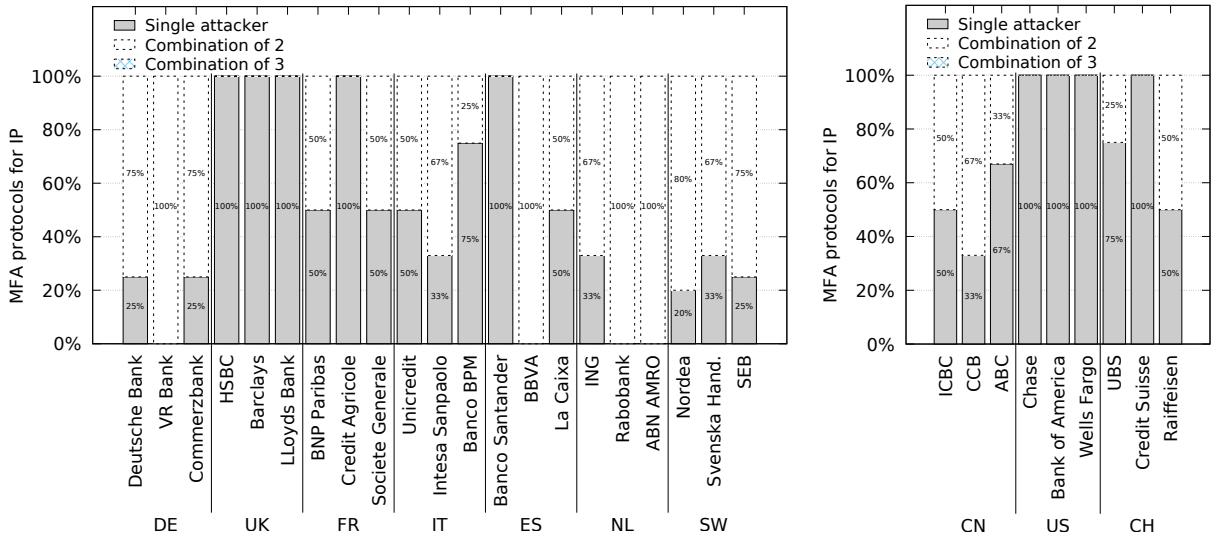
Globally, we observe that the considered best practices are fulfilled, on average, by more than a half of the EU banks. For non-EU banks the level of compliance is lower. The most violated best practices are **BP2** and **BP7**. The first one is rarely fulfilled because almost every application released by banks relies on proprietary or commercial solutions, rather than the APIs we identified in Table 4.17. In the case of **BP7**, the lack of fulfillment is due to the fact that, if two physical authenticators are offered, one of the two is usually given upon request and payment of a little sum of money. On the other hand, the most fulfilled best practices are those related to the perceived ease-of-use of the digital authentication, namely **BP3** and **BP8**. Indeed, as seen in Section 6.2.3.1, the majority of the banks employs both exemptions and a high variety of authenticators.

6.2.3.3 Resistance to attacker models

In this section, we discuss how the MFA protocols adopted by banks behave with respect to the attacker models described in Section 4.1.1. Here, we provide an overview of the results and only report when an MFA protocol can be successfully compromised by one of the attacker models individually or only by their combination. We refer to the supplementary material for a detailed evaluation of MFA protocols (e.g., which and how many attackers that can compromise a protocol by acting individually).

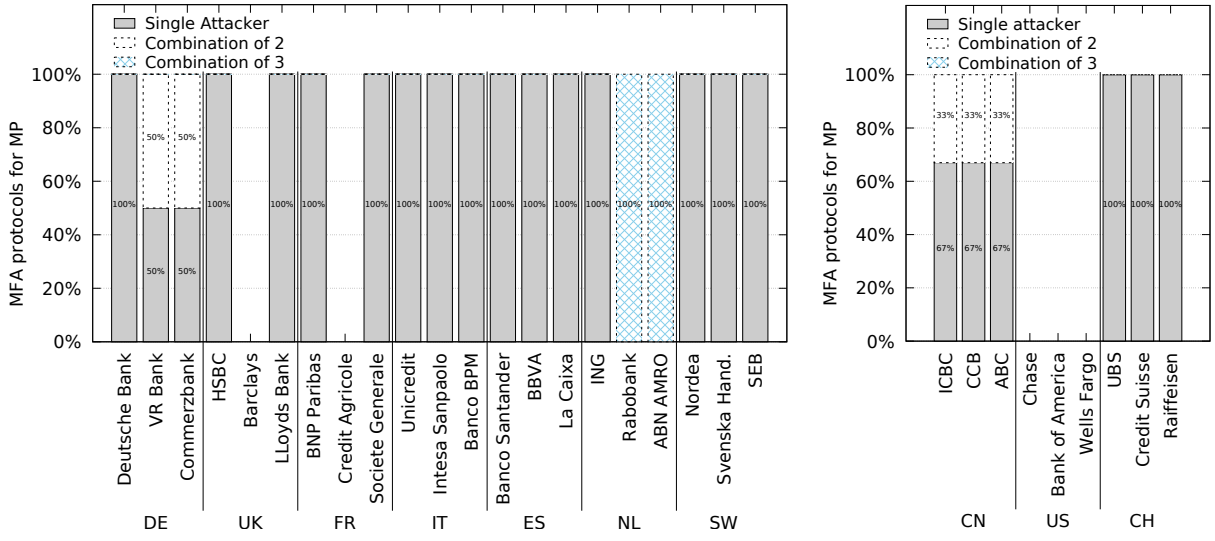
Figure 6.10 shows, for each bank, the percentage of MFA protocols for IP that can be compromised by single attacker models and their combinations (composed either by two or three attacker models), whereas Figure 6.11 shows the results for MFA protocols for MP. The percentage of MFA protocols that are vulnerable to attacker models acting individually is represented by solid gray boxes, whereas the percentage of MFA protocols that are only vulnerable to combinations of attacker models is represented by white (two attacker models) or light blue (three attacker models) pattern-filled boxes with dashed lines. Trivially, MFA protocols that are vulnerable to single attackers, are also vulnerable to their combination with other attacker models. We refer to Section 6.2.3.1 for the number of MFA protocols for IP and MP offered by each bank.

We observe that 46% (on average) of the MFA protocols for IP adopted by each EU bank are vulnerable to single attacker models. In particular, at least half of the MFA protocols offered by 10 EU banks are vulnerable to single attacker models. All MFA protocols offered by 5 banks (all English, one Spanish and one French) can be compromised by single attacker models. Non-EU banks offer an average of 62% of the MFA protocols for IP that can be compromised by single attacker models. It is worth noting that the MFA protocols offered by all US banks and by one Swiss bank are vulnerable to single attacker models.



(a) EU banks (b) non-EU banks

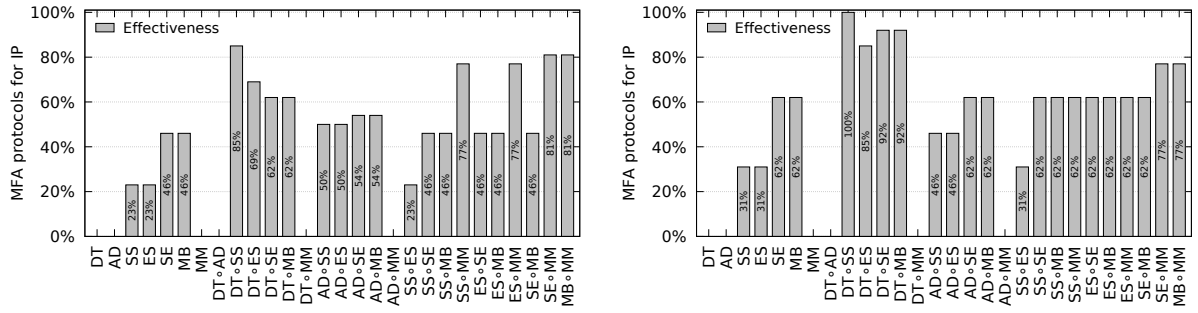
Figure 6.10: Percentages of MFA protocols for IP vulnerable to single or combined attackers.



(a) EU banks (b) non-EU banks

Figure 6.11: Percentages of MFA protocols for MP vulnerable to single or combined attackers.

In the context of MP, the percentage of vulnerable protocols is higher. 85% (on average) of the MFA protocols offered by each EU bank are vulnerable to single attacker models. Only 6 EU banks offer at least one MFA protocol that cannot be compromised by single attacker



(a) EU banks

(b) Non-EU banks

Figure 6.12: Percentage of MFA protocols for IP that are vulnerable to given attackers.

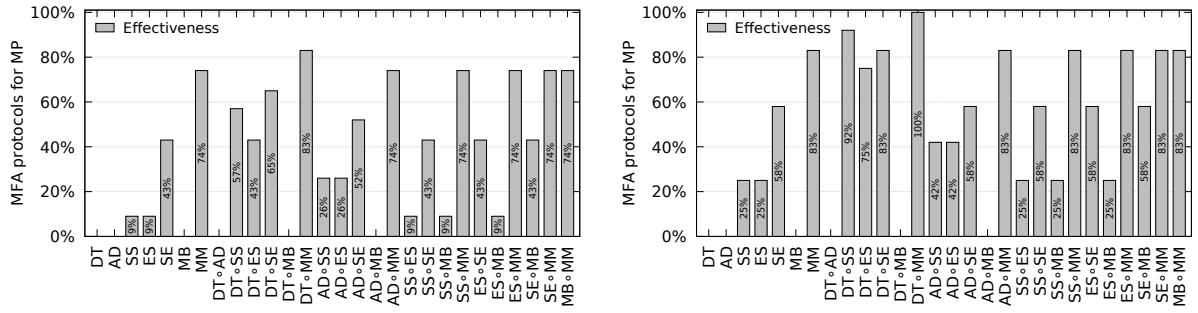
models, but only by their combination. It is worth noting that the missing box for Barclays and Credit Agricole is due to the fact that these banks do not provide any MFA protocol for MP (see Figure 6.4). On the other hand, 83% (on average) of the MFA protocols offered by each non-EU bank are vulnerable to single attacker models. US banks do not provide any MFA protocol for MP, and only the Chinese banks offer at least one MFA protocol that cannot be compromised by single attacker models.

Interestingly, our analysis revealed that 97% of all MFA protocols (both for IP and MP) can be compromised by at least one combination of two attacker models. The other 3% (2 MFA protocols for MP offered by Rabobank and ABN Amro) require a combination of at least three attacker models to be compromised.

We now present an analysis of the effectiveness of different attacker models over the employed MFA protocols (see Section 4.1.1 for the details on each attacker model). Figure 6.12 and 6.13 show the percentage of MFA protocols (for IP and MP, respectively) that can be compromised by single attacker models or by combinations of two of them. The effectiveness of single and composed attacker models is represented by a gray box.

From the figures, we observe that the most effective attacker models against MFA protocols for IP are Man in the Browser (MB) and Social Engineer (SE). When taken individually, these attacker models can compromise 48% and 67% of the MFA protocols for IP employed by EU and non-EU banks, respectively. On the contrary, Man in the Mobile (MM), Device Thief (DT) and Authenticator Duplicator (AD) alone are never able to compromise any MFA protocol for IP. If we consider combinations of attacker models, the most effective combination of two attacker models on the MFA protocols for IP employed by EU banks is constituted by DT and Shoulder Surfer (SS), being able to compromise 84% of these protocols. In the case of non-EU banks, the combined attacker model can potentially compromise all adopted MFA protocols for IP.

For what concerns MFA protocols for MP, Man in the Mobile (MM) is the most effective attacker model. Indeed, it can compromise – by itself – 74% and 83% of the MFA protocols for



(a) EU banks

(b) Non-EU banks

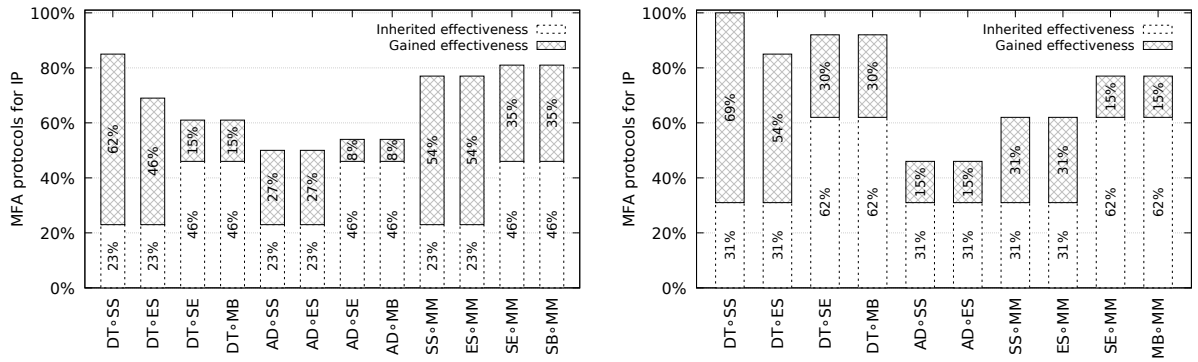
Figure 6.13: Percentage of MFA protocols for MP that are vulnerable to given attackers.

MP offered by EU and non-EU banks, respectively. The most effective combination of two attacker models is "DT∘MM", managing to compromise 83% and 100% of the protocols offered by EU and non-EU banks, respectively. Also in the context of MP, DT and AD are not effective against any MFA protocol when acting by themselves. In particular, these attacker models compromise the ownership factors asserted by an authenticator, but they are not able to compromise knowledge factors, which are used in all analyzed MFA protocols.

We stress that there is no combination of two attacker models that is able to compromise all MFA protocols (either for IP or MP) offered by EU banks. The minimum number of attacker models required to achieve this result is 3. In particular, only a combination of DT, SS and MM, is able to compromise all MFA protocols (either for IP or MP) offered by EU banks.

We now analyze the effectiveness of combinations of attacker models. In particular, we assess the effectiveness "gain" that combinations of attacker models have with respect to the effectiveness of the attacker models that they include. The gain (represented with a box filled with gray pattern) is the percentage of protocols that can be compromised only by exploiting the capabilities of all attacker models in the combination. Moreover, the figures show the "inherited" percentage of effectiveness, i.e., the percentage of protocols compromised by the attacker models in the combination when acting individually (represented with a white box with dashed line). In the case an MFA protocol can be compromised by more than one attacker model, it is considered only once in the computation of the inherited value.

Consider, for instance, combination "DT∘SS∘SE". According to Figure 6.15a, this combination has 78% of inherited effectiveness and 14% of gained effectiveness. The inherited effectiveness is obtained by considering the protocols that can be compromised by DT, SS or SE individually or by all combinations including two of these attacker models. If an MFA protocol is compromised by more than one of these (composite) attacker models, it is counted only once for assessing the inherited effectiveness. As shown in Figure 6.13a, "DT∘SS" has 57% of effectiveness, "DT∘SE" has 65% and "SS∘SE" has 43%. However, "DT∘SE" compromises all MFA



(a) EU banks

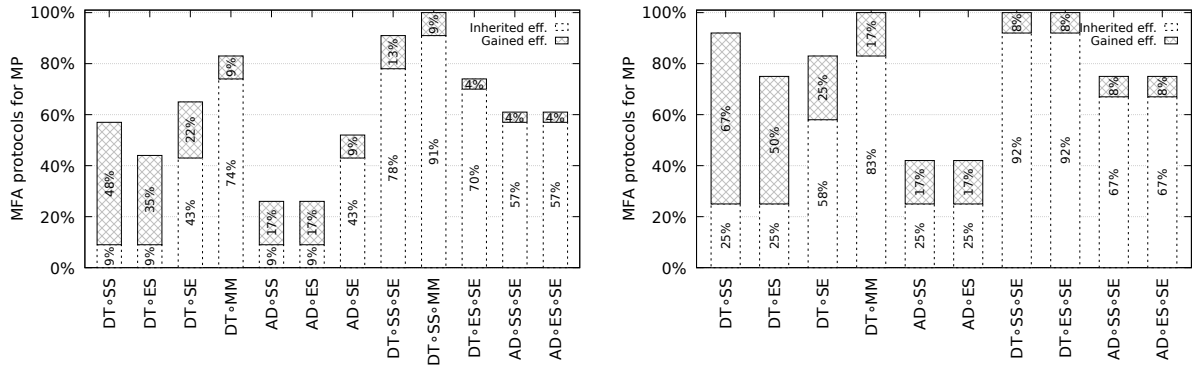
(b) Non-EU banks

Figure 6.14: Percentage of MFA protocols for IP that are vulnerable to given attackers.

protocols compromised by "SS∘SE" and the majority of those compromised by "DT∘SS", managing to compromise only 23% additional MFA protocols in respect with "DT∘SS". Therefore, the inherited effectiveness is 78%. The gained effectiveness, instead, derives from the number of MFA protocols that can only be compromised by DT, SS and SE acting together (i.e., that cannot be compromised by any of the two combinations). In this case, "DT∘SS∘SE" (acting together) can compromise 13% of the MFA protocols in addition to the 78% compromised by "DT∘SS" and "DT∘SE", obtaining a total effectiveness of 91%.

Figures 6.14 and 6.15 present the results of our analysis. Note that combinations not having any gain with respect to the effectiveness of the attacker models that they include are not shown in the figures. We observe that the most effective combination (i.e., with higher gain) is "DT∘SS". In general, these two plots show that the most effective combinations are those combining attacker models having different targets (in terms of authenticators, authenticator outputs and authentication factors). The "DT∘SS" combination, for example, includes DT – that targets ownership factors – and SS – targeting knowledge factors and manually copied `otps`). This result is not surprising, since MFA protocols should be designed in such a way that a potential attacker is required to execute multiple (and different) malicious actions in order to compromise the protocol.

To conclude, we observe that the least secure MFA protocols, both for IP ad MP, are those employing authenticators generating an `otp` without receiving an `opid` as an input. Indeed, as shown in the supplementary material, these protocols are the most vulnerable ones in terms of resistance to singletons and to combinations of attacker models.



(a) EU banks

(b) Non-EU banks

Figure 6.15: Percentage of MFA protocols for MP that are vulnerable to given attackers.

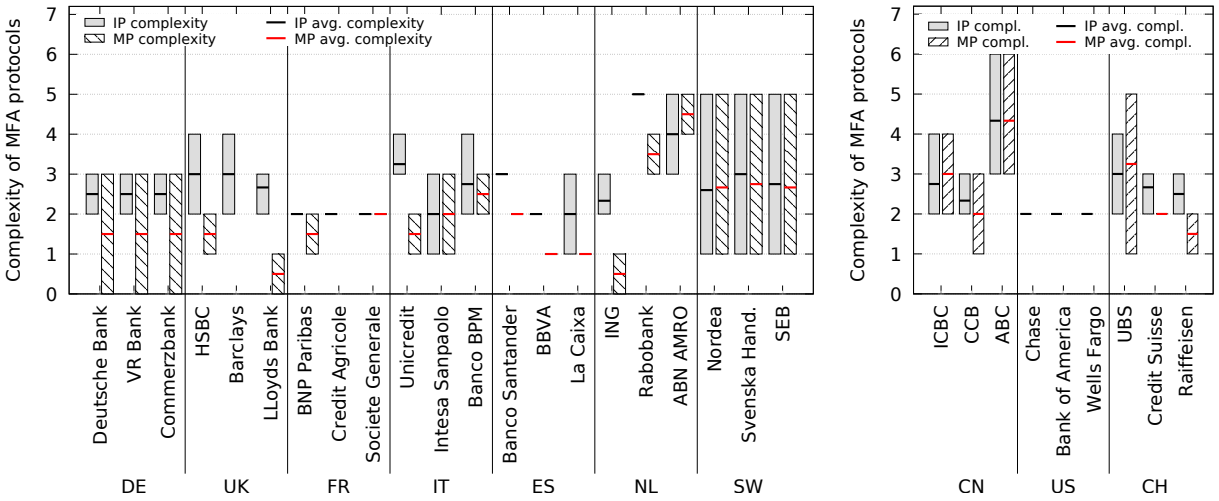
6.2.3.4 Complexity of MFA Protocols

In this section, we analyze the complexity of the MFA protocols adopted by banks against the criteria defined in Section 4.3. We first compute the complexity score[8] of the MFA protocols adopted by each bank and then investigate to what extent the various types of resources affect the overall complexity of MFA protocols. A detailed analysis of the complexity of the MFA protocols is given in the supplementary material.

Figure 6.16 shows the complexity score of the MFA protocols for IP and MP (represented by gray and pattern-filled boxes, respectively) employed by each bank. In the plots, we represent the average complexity score of the MFA protocols for IP and MP employed by each bank using a black and red line respectively, where the average is computed over the number of MFA protocols employed by each bank as reported in Section 6.2.3.1. From the figure, we observe that the average complexity for each bank is homogeneous between banks of the same country, with the exception of the Netherlands. In particular, the difference in the complexity of MFA protocols adopted by banks in the same country never exceeds 1.6 (except for the Netherlands, where this difference is 2.67 and 4 for IP and MP, respectively). The difference observed in the Netherlands is mainly due to a single bank, i.e., ING, which offers its customers a set of MFA protocols that notably differ from those offered by the other banks in the country. Given the small number of banks considered for each country, we cannot determine to what extent this result represents a national trend (see further discussion on this point in Section 6.3).

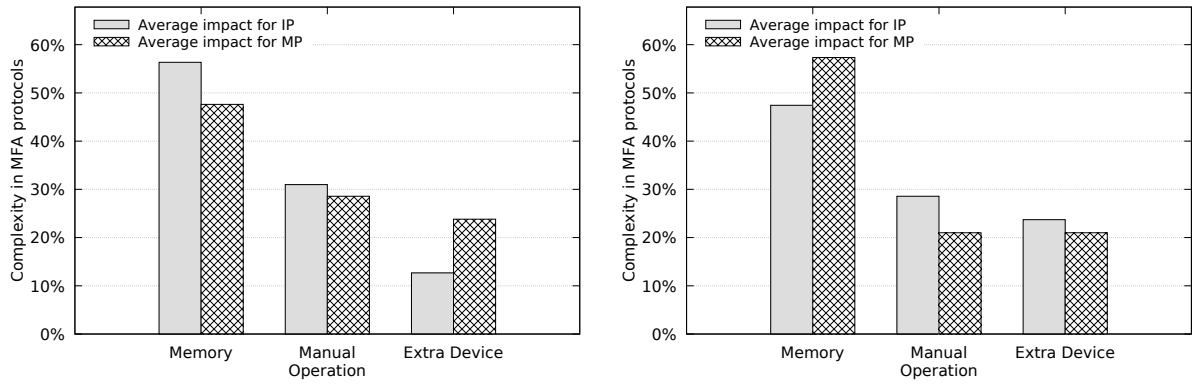
Moreover, we observe that MFA protocols for IP are, in general, more complex than those for MP. This fact is particularly noticeable for the MFA protocols employed by EU banks where the average complexity of MFA protocols for IP is 2.8 and the average complexity of MFA protocols

[8]Recall from Section 4.3 that the complexity score of a MFA protocol is computed by summing up the amount of resources – memory, manual operations and extra devices – required by the protocol.



(a) EU banks (b) non-EU banks

Figure 6.16: Complexity of MFA protocols adopted by banks.

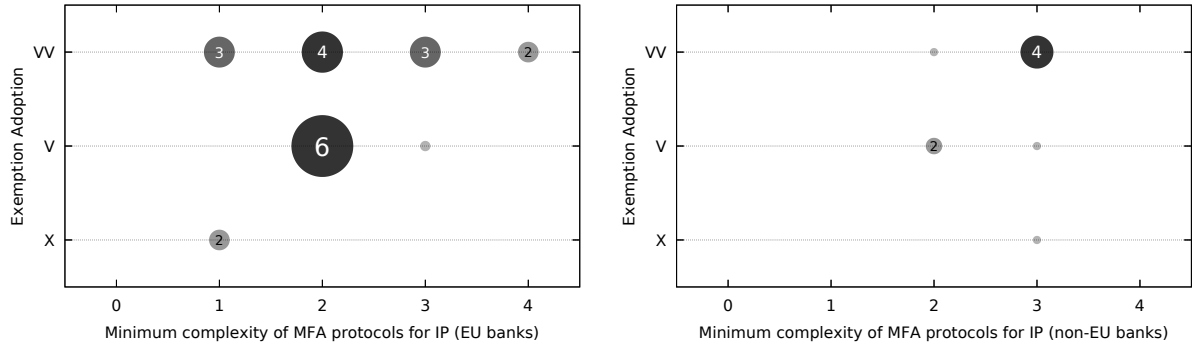


(a) EU banks (b) Non-EU banks

Figure 6.17: Average complexity of MFA protocols.

for MP 2.1. On the other hand, this difference is lower for non-EU banks (3.5 against 3).

Figure 6.17 shows the impact of the different types of resources – memory, manual operations and extra devices – on the overall complexity of MFA protocols both for IP and MP. We observe that memory efforts have typically a higher impact on the overall complexity of MFA protocols compared to the other two complexity aspects (around 50% of the complexity score). This is because the majority of the MFA protocols offered by banks rely on at least one knowledge factor and more than 30% of MFA protocols leverage two of them (see Section 6.2.3.1). On the other hand, the average complexity deriving from bringing an extra device is usually lower than the



(a) EU banks (b) Non-EU banks

Figure 6.18: Relationship between exemptions and complexity (IP).

other two complexity aspects.

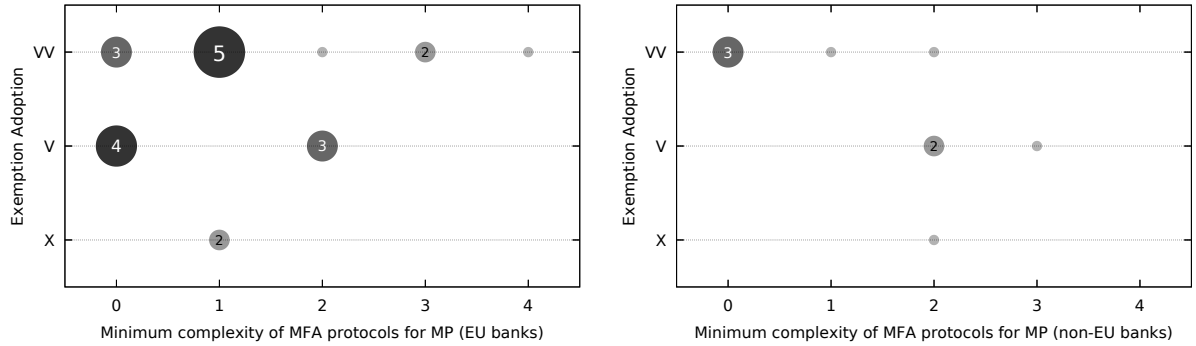
An in-depth analysis reveals that MFA protocols with the lowest complexity are the ones for MP that only leverage combinations of inherence (i.e., fingerprints) and ownership factors and that do not require users to bring any extra device, any memory effort or manual input. On the contrary, the most complex MFA protocols are the ones involving at least one knowledge factor and a multi-factor hardware authenticator that requires users to manually insert `opid` and manually copy the obtained `otp` into the endpoint.

6.2.3.5 Correlations

In our analysis, we also investigated on possible correlations between the obtained results. In particular, we scrutinize on possible correlations between compliance (with requirements and best practices), robustness against security threats and complexity of the MFA protocols adopted by banks.

6.2.3.5.1 Correlations between exemptions and complexity As a first investigation, we hypothesized that there exists a correlation between the adoption of exemptions and the complexity of MFA protocols employed by banks.

The results of our analysis is presented in Figures 6.18 and 6.19, which show the number of banks employing exemptions and the minimum complexity of the adopted MFA protocols for IP and MP, respectively. From three of these figures, we observe the lack of correlation between these two aspects. Figure 6.19b, instead, seem to show an inverse correlation. This is confirmed by the Pearsons's correlation coefficients. For EU banks, we obtained coefficient of 0.37 and 0.19 (for IP and MP, respectively). For non-EU banks, we obtained a coefficient of 0.11 for IP



(a) EU banks

(b) Non-EU banks

Figure 6.19: Relationship between exemptions and complexity (MP).

and -0.64 for MP.

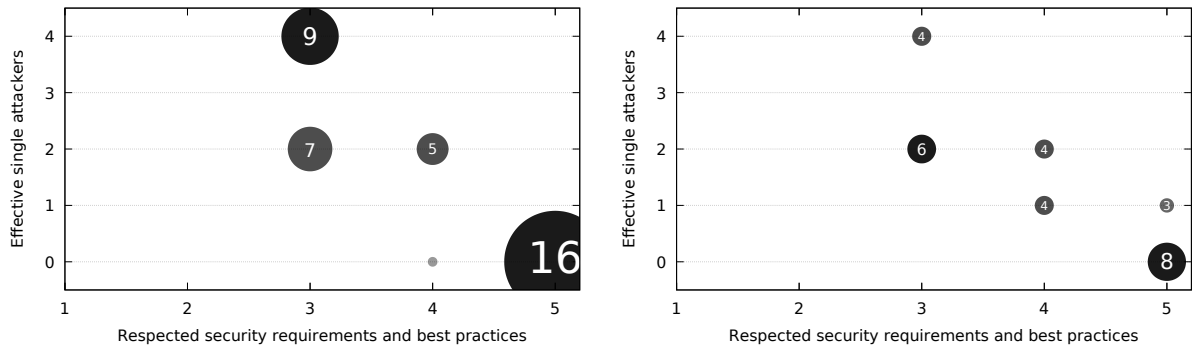
To assess if our results are statistically significant, we used the Fisher's exact test. The obtained p-values for EU banks are 0.073 and 0.066 for IP and MP respectively, whereas for non-EU banks they are 0.64 and 0.40 for IP and MP, respectively. Since the p-values are higher than the significance level of 5%, we cannot reject the null hypotheses.

Therefore, our first hypothesis is not supported by the results. We conclude that the adoption of exemptions (hence the compliance with **BP3**) – even if very frequent – might not be related to the employment of complex MFA protocols.

6.2.3.5.2 Correlation between compliance to security requirements (and best practices) and resistance. The second hypothesis aims to assess the effectiveness of security requirements and best practices on the robustness of MFA protocols against attacker models. To this end, we verify whether there exists a correlation between the compliance of an MFA protocol with requirements and best practices (related to security aspects) and its resistance against the attacker models presented in Section 4.1.1.

Figure 6.20 presents the results of the analysis by indicating the number of MFA protocols (both for IP and MP) that comply with requirements and best practices and are vulnerable to single attacker models. The level of compliance with the aforementioned security requirements and best practices is computed as the number of requirements and best practices met by the protocol.

It is interesting to observe that all MFA protocols comply with at least 3 security requirements and best practices. Furthermore, we observe that the more security requirements and best practices are met by an MFA protocol, the more robust the protocol is against attacker models. This is especially evident in the case of MFA protocols for IP: 16 MFA protocols comply with all considered security requirements and best practices and none of them is vulnerable to single



(a) MFA protocols for IP

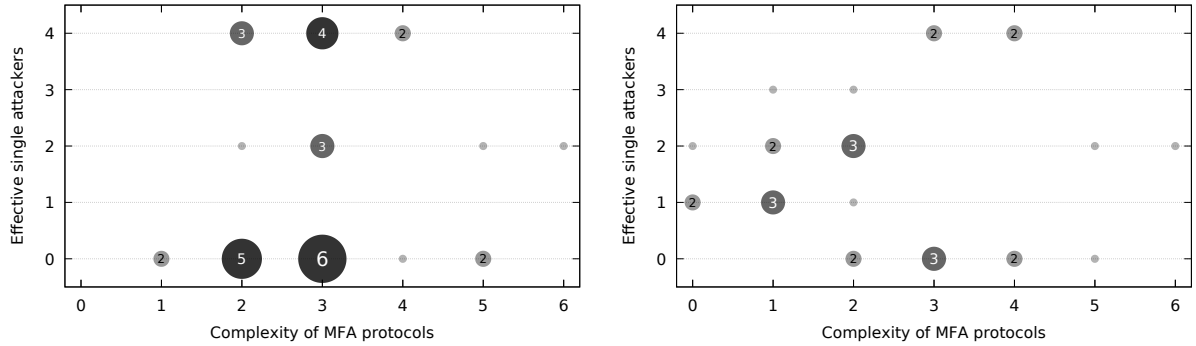
(b) MFA Protocols for MP

Figure 6.20: Relationship between compliance with security requirements (and best practices) and resistance to attacker models.

attacker models. On the other hand, almost all MFA protocols complying with less than four security requirements and best practices are vulnerable to at least two single attacker models. A similar trend can also be observed in the case of MFA protocols for MP. The Pearsons's coefficient confirms our intuition. Specifically, the Pearsons's coefficient is -0.93 and -0.84 for MFA protocols for IP and MP, respectively. These values show the presence of a very strong inverse correlation between the two variables.

To investigate the most relevant security requirements in the correlation, we calculate the correlation between the resistance of MFA protocols against attacks (i.e., the number of singletons) and every security requirement and best practice in the set. We notice that **RL6** and **RL5** are the requirements that mainly influence the robustness of MFA protocols against attacker models. In particular, the correlation between **RL6** and the number of attackers compromising an MFA protocol is 0.90 and 0.82 (for IP and MP, respectively), while between **RL5** and the number of attackers is 0.86 and 0.73 (for IP and MP, respectively). These results show that the use of operation-dependent `otp` and keeping the user aware of the operation she is going to authorize are the more effective solutions against the identified attacker models. We performed the Fisher's exact test to verify the statistical significance of our findings. The obtained p-values are $5.93e{-}10$ and $6.08e{-}7$ (for IP and MP, respectively), which are lower than the fixed significance level of 5%. Therefore, the null hypotheses can be rejected and our results can be considered to be statistically significant.

We can conclude that this hypothesis holds, indicating that the compliance with security requirements and best practices increases the resistance of MFA protocols to the considered attacker models.



(a) MFA protocols for IP

(b) MFA Protocols for MP

Figure 6.21: Relationship between complexity and resistance to attacker models.

6.2.3.5.3 Correlation between complexity and resistance to attackers. The third hypothesis aims to assess the independence between the complexity of an MFA protocol and its resistance against individual attacker models.

Figure 6.21 shows the results of the analysis for MFA protocols both for IP and MP. We note that there is not a clear correlation between the complexity of an MFA protocol and its resistance against attacks in both IP and MP context. An example of this can be observed for MFA protocols with complexity equal to 3. We can find both MFA protocols resistant against every individual attacker model and MFA protocols vulnerable to 4 of them.

We compute the Fisher's exact test to assess the independence between the complexity score of an MFA protocol and its resistance to attacker models. The obtained p-values for the IP and MP cases are 0.83 and 0.44, respectively. Both the values are higher than the significance level of 5%. Hence, we can confirm that the two variables are independent.

Therefore, we can conclude that a complex MFA protocol is not necessary more resistant against attacker models.

6.3 Threats to Validity and Generality

In this section we list the limitations of our study and discuss their potential impact on the validity of our work. We distinguish between four types of threats, namely *internal*, *external*, *construct* and *conclusion*. We also discuss to what extent our methodology can be generalized to other application domains.

6.3.1 Internal threats

Internal threats to validity are mostly related to our bank and, thus, MFA protocol dataset. We obtained all information relevant and necessary for the analysis from the documentation, tutorial and demos made available by banks. In fact, banks tend to publish as much documentation as possible to help their customers in the use of their services. However, as mentioned in Section 6.2.1, we had no direct interaction with banks and we only accessed public information and documentation. Therefore, we did not have access to information concerning the server side of the MFA implementation or technical features of the employed authenticators. These technical features could have effects that can be only partially captured by our analysis.

Moreover, we did not consider the release time and evolution of MFA protocols in our dataset. Reasonably, an MFA protocol should be evaluated against the regulations and directives that were in force when it was released. The changes in the legal framework (see Section 4.2.1) might have resulted in security patches and updates of older MFA protocols and older protocols might be supported for backward compatibility. In this case, banks might require new customers to only use the newer MFA protocols and even force old customers to use them.

6.3.2 External threats

The main external threat to the validity of our study is related to the size and composition of our dataset. Since we only consider three banks per country, we cannot fully support statements on national trends. Moreover, we selected banks according to their dimension. Arguably, large banks have more resources to invest on the security of their services. Thus, extending our analysis to smaller banks would result in a more precise characterization of the online banking landscape.

While MFA has been adopted in several types of online services, our analysis only focuses on MFA implementations adopted for online banking. Different types of online services can have very different business models and requirements. These differences can have a significant impact also on the employed MFA implementations. In this respect, our findings cannot be generalized to other types of online services. Below (generality), we discuss to what extent our methodology can be used for the analysis of MFA implementations adopted in other application domains.

6.3.3 Construct threats

A potential construct threat is our interpretation of regulations, directives and best practices. As a matter of fact, some definitions and descriptions contained in the documentation are informal or vague. This is probably done on purpose to make the rules generic and widely applicable. From our perspective, we had to cast these concepts to more rigorous and precise definitions. A concrete example is the interpretation of **BP2**, where we provided a list of standard solutions.

Although based on standards defined by manufacturers, such a list might be too restrictive. For instance, some commercial solutions are commonly adopted and they represent the de facto standard.

Another construct threat is related to our representation of MFA protocols. In this work, we reconstructed MFA protocols by observing the client side of the authentication process. Since we have no information concerning the sequence of operations performed on the server side, we assume that such operations are executed properly and the communications between server and client work as intended. Moreover, we abstracted some details regarding the sequence of messages exchanged during protocol execution. This lack of information has also an impact on our representation of authenticators. Although authenticators are, by definition, used on client side, their behavior might depend on some procedure executed remotely, e.g., the generation of authentication tokens. For instance, we did not consider the possible impairment of keys and seeds for the `otp` generation on the server side. Overall, the analysis of a low-level implementation might be subject to additional security risks and considerations.

6.3.4 Conclusion threats

In the analysis of the correlation between different criteria, the validity of our conclusions were evaluated in terms of the statistical significance of the obtained results. In some cases, e.g., for hypothesis *H1*, the null hypotheses cannot be discarded and, thus, our conclusions cannot be considered statistically significant.

6.3.5 Generality

In this work, we focused on the MFA implementations used by some online banking services. Nevertheless, our approach for analyzing the robustness and complexity of MFA protocols is independent from the specific application domain. Therefore, our approach can be applied to analyze MFA protocols in general. On the other hand, assessing the compliance with laws and regulations is context-dependent. For instance, the European laws for digital identity (e.g., eIDAS [Eur14]) do not require "dynamic linking", which is a critical factor for the online banking sector [Eur15, Eur17]. Therefore, a shift in the application domain would require rethinking the evaluation of the compliance with laws and best practices.

Chapter 7

Related Work

In this chapter, we firstly present the related work concerning the analysis of MFA protocols. Then, we compare our research in the field of online banking with other surveys.

7.1 Analysis of MFA protocols

Several publications on the analysis of security protocols have been released in the last decades. Among them, formal methods have a central role and they have been used to discover several flaws in a number of protocols (e.g., [Low96, APS14, BV11, BM11, CGDW09])). As a consequence, many formal verification frameworks have been put forward [Vig06, MSCB13, Bla14, ACC09]. Nevertheless, only some frameworks targeting MFA protocols have been proposed.

Some authors applied formal methods also for the analysis of MFA protocols. In [JK18], the authors propose an analysis framework for evaluating MFA protocols against a set of adversaries, including both the Dolev-Yao and new adversaries based on common human errors. In particular, each protocol is modeled as an agent of the pi-calculus. The pi-calculus specification is then refined by considering certain factors such as common human errors. Finally, the specification is given as input to the PROVERIF [Bla14] tool for the analysis. Some of the threat models of [JK18] are in common with our work (e.g., the Eavesdropping Software), but they do not consider others (e.g., Device Thief) that we consider here. Moreover, there are several other differences. First of all, [JK18] does not support the concept of authenticator. Instead, their modeling technique requires specific internal information about the protocol. In many real cases, this is unlikely to happen. On the other hand, [JK18] model common human errors that we do not consider in this thesis.

In [SCRV18, SCRV], the authors present the design of a multi-factor authentication solution for native mobile apps, along with a formal specification and analysis of the presented protocol. In

particular, the authors propose a novel design of protocol for performing MFA with a Single Sign-On capability on a mobile device. The design is then formally specified using ASLAN++ [vOM12], a high-level, role-based formal language supporting the specification of different variants of authentication and confidentiality properties. Such a formal model is then analyzed with the SATMC [ACC16] model checker, one of the model checkers of the AVANTSSAR platform [AAA+12]. The security analysis is performed against a Dolev-Yao attacker model, to which are progressively added capabilities by removing different kind of security assumptions characterizing various protocol aspects. Although they also consider an attacker model inspired to [NIS17], [SCRV18] has a different focus. In particular, they focus on the fine-grained, formal verification of a single protocol, while in this work we prioritize the number of protocols. Moreover, our technique requires less information for the MFA protocol specification, thus making our methodology applicable in more contexts.

In [PRW17], the authors perform a security analysis on the FIDO U2F, leveraging pi-calculus for modeling the protocol. The evaluation is performed against an attacker that has control over the network and it can either attack by itself or relying on a dishonest Server or Client. The analysis highlighted how the verification of the appID parameter is key for the security. In our work, we perform a different kind of analysis, assuming the Server and the Client are acting honestly and all the checks on values are performed.

In [ACZ13], the authors performed a formal analysis on two Secure Call Authorization protocols. These protocols feature use two-factor and two-channel authentication The authors found a vulnerability on the protocols by evaluating them against Dolev-Yao attacker and Social Engineer. Again, the methodology proposed in [ACZ13] requires details about the protocol that are not necessary for our technique. Many other authors followed simiral approaches, e.g., [DeF11, Hag07, JCL+17, JJIL16]. In many cases, these authors could spot out actual vulnerabilities in some specific MFA protocols. Yet, all of them rely on protocol internals as discussed above.

In [Gar16], the author applies static vulnerability analysis on four MFA protocols, employing passwords and smart cards or secure storage. These protocols are evaluated in terms of resistance against the clogging attack, a form of denial of service. The analysis of [Gar16] is performed on cryptographic operations and primitives, as modular exponentiation. Various vulnerabilities on the analyzed protocols have been found and countermeasures have been proposed. This kind of analysis is out of the scope of our work, since we assume perfect cryptography and we do not model the internal operations of the authenticators employed in a protocol.

In [OBM+18], the authors provide a survey on MFA in terms of the available sensors and factors that can be employed in an MFA implementation. The authors present a plethora of different types of sensors, comparing them according to different criteria. Moreover, the authors presenting the open challenges in terms of security, technological perspective and user experience. Finally, the authors propose a framework for utilizing MFA in cases where some of the factors are missing, guaranteeing a proper level of security and usability. This work gives multiple insights

Table 7.1: Comparison with other surveys on protocols for online banking.

		Choubey et al. [CC13]	Kiljan et al. [S. 16]	Dmitrienko et al. [DLRS14]	Krol et al. [KPCS15]	Althobaiti [Alt16]	Our survey
Dataset	Number of Banks	60	80	4	10	–	30
	Geographical Distribution	UK,CA,IN,IRL US,RSA,AU	World	DE	UK	SA,UK	DE,UK,FR,IT,ES, NL,SW,CN,US,CH
	Number of MFA Protocols	–	80*	6	9	3	61 (153)
	Endpoints	IP	IP, MP†	MP	IP	IP	IP, MP
	Temporal evolution	–	✔	–	–	–	–
	Authentication factors/authenticators	✔	✔	–	✔	✔	✔
	Enrollment and Binding	–	–	–	–	–	✔
Compliance	Regulations	–	–	–	–	–	✔
	Best Practices	–	–	–	–	–	✔
Security	Security of TLS/SSL implementation	–	✔	✔	–	–	–
	Perceived security	–	–	–	✔	✔	–
	Resistance of MFA protocols against attacker models	–	–	✔	–	–	✔
Usability	Perceived usability	–	–	–	✔	✔	–
	Complexity	–	–	–	–	–	✔
Correlations	Exemptions and Complexity	–	–	–	–	–	✔
	Compliance with security requirements and resistance to attackers	–	–	–	–	–	✔
	Complexity and resistance to attackers	–	–	–	–	–	✔

* No reference to unique MFA protocols.
† Only classification of authenticator factors.

of the various aspects of MFA. Nevertheless, since the authors do not put forward a methodology for the analysis of existing protocols, their proposal covers a different aspect w.r.t our work.

7.2 Surveys on MFA in Online Banking

Comparison and surveys has initially been published only by companies proposing new MFA solutions or generic security products (as [Cen16] [Gem15] [Pin09]). In the following years, new surveys on MFA protocols have been published. However, the majority of such publications is focused on the specific application scenario.

For what concerns the banking sector, for instance, new designs of several MFA protocols for online banking have been summarized in a few surveys. These surveys usually provide a classification and a comparison of MFA protocols and implementations. Choubey et al. [CC13] analyze the authentication mechanisms for IP employed by banks of 7 countries. In particular,

the authors provide a classification of the adopted authenticators and emphasize the lack of a standardization in the design of MFA protocols. Kiljan et al. [S. 16] review the authentication and communications protocols for online banking adopted by 80 banks worldwide. This study provides an analysis of the temporal evolution of MFA protocols adopted by banks, together with a classification of the used authentication factors and MFA protocols for both IP and MP. The security of MFA protocols for IP is evaluated by analyzing the implementation of the underlying TLS/SSL mechanisms whereas the security of MFA protocols for MPs is not analyzed. However, no evaluation regarding the resistance to attacker models and compliance with regulations (and best practices) is provided. Dmitrienko et al. [DLRS14] analyze the security of 6 commonly used MFA protocols for MP. In particular, they identify the main weaknesses of these MFA protocols in terms of potential implementation errors and resistance to attacker models. No data regarding usability and MFA implementations adopted by banks is presented. Krol et al. [KPCS15] analyze the usability and perceived security of the authentication mechanisms employed by 10 UK banks (for a total of 9 MFA protocols) through user studies. Similarly, Althobaiti [Alt16] evaluates the security and usability of MFA protocols based on questionnaires and field tests.

The aforementioned studies differ from each other for the analyzed features and scope. Table 7.1 summarizes the main differences between those studies and our work. A primary difference is in the analyzed dataset and, in particular, in the number of banks and MFA protocols considered. All surveys provide an analysis of MFA protocols along with the used authenticators, with the exception of the work in [CC13], which only provides a classification of authenticators without analyzing the protocols in which they are used. However, most surveys only analyze MFA protocols specific to one endpoint, with the majority considering protocols for IP. Our survey considers protocols for both IP and MP, since they might provide different security levels and user experience. Existing surveys also do not consider user enrollment and the binding of authenticators. Nevertheless, these phases can affect the overall security of an MFA protocol. Moreover, none of the previous works assesses the compliance of MFA solutions with laws and best practices. We claim that this aspect is also relevant, since often laws and best practices define a baseline for the security guarantees that an MFA protocol must provide. Security aspects of MFA protocols are considered by most surveys, but at a different level compared to our work. For instance, some surveys [S. 16, DLRS14] analyze weaknesses in MFA implementations, whereas others [KPCS15, Alt16] focus on the security of MFA protocols perceived by users. In contrast to these studies, our work evaluates the security of MFA protocols by assessing their robustness against some attacker models. This analysis aims to compare MFA protocols in terms of resistance to well defined attack scenarios. At the best of our knowledge, the only other proposal considering an attacker model for MFA protocols is [DLRS14]. However, their attacker model only considers the MP context. Moreover, only a few surveys [KPCS15, Alt16] evaluate the usability of MFA protocols. However, differently from those surveys that evaluate perceived usability and user satisfaction of MFA protocols through user studies, we focus on the efficiency of MFA protocols and propose an "objective" measurement of the complexity of MFA protocols, which can be computed from the dataset at hand. Finally, our survey is the only one that aim to discover

correlations between compliance with laws and best practices, security and usability aspects. Leveraging this investigation, we are able to verify how the different features of MFA protocols and their compliance with laws and best practices are realized along with their effects.

Chapter 8

Conclusion

In this thesis, we proposed a framework to analyze MFA protocols relying on neither implementation details nor behavioral specifications. To this aim, we defined a specification language for modeling MFA protocols even having a limited amount of details. In practice, it is possible to model a protocol in terms of the operations that a user is required to perform during an authentication session. In particular, the presented language focuses on the concept of authenticator as the basic building block of any MFA protocol. The semantics of the proposed language is given in *Security Protocol Specification* (SPS) language [AMV15].

For what concerns the security analysis, MFA protocols are evaluated in terms of resistance against a set of attacker models. These attackers have been tailored for the specific case of MFA protocols, being modeled following dedicated literature (as [JFR17, Ayy17, LM11]) and NIST definitions [NIS17]. For what concerns the regulatory aspects and best practices, MFA protocols are evaluated in terms of compliance with a customizable set of requirements and best practices. In this work, we extracted and encoded a set of requirements (deriving from EBA regulations) and a set of best practices (derived from [NIS17, PCI16a] and others) and use them for evaluating a protocol. Furthermore, a new metric, called *complexity*, is employed for evaluating a protocol in terms of efforts that a user is required to perform during its execution.

The framework has been then implemented in a working tool, MuFASA. By showing an user-friendly interface, MuFASA allows (even non-expert users) to model an MFA protocol and to automatically analyze it, giving a report presenting the performed evaluations and the obtained results.

Finally, the presented framework has been applied on some selected use cases. First, it has been employed in the early stages of a new MFA protocol design. Then, it has been used for performing a latitudinary study on online banking services, allowing us to model and analyze more than 150 MFA protocols employed by banks all over the world.

Bibliography

[AAA+12] Alessandro Armando, Wihem Arsac, Tigran Avanesov, Michele Barletta, Alberto Calvi, Alessandro Cappai, Roberto Carbone, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, Gabriel Erzse, Simone Frau, Marius Minea, Sebastian Mödersheim, David von Oheimb, Giancarlo Pellegrino, Serena Elisa Ponta, Marco Rocchetto, Michael Rusinowitch, Mohammad Torabi Dashti, Mathieu Turuani, and Luca Viganò. The avantssar platform for the automated validation of trust and security of service-oriented architectures. In Cormac Flanagan and Barbara König, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 267–282. Springer Berlin Heidelberg, 2012.

[ACC09] Alessandro Armando, Roberto Carbone, and Luca Compagna. Ltl model checking for security protocols. *Journal of Applied Non-Classical Logics*, 19(4):403–429, 2009.

[ACC16] A. Armando, R. Carbone, and L. Compagna. Satmc: a sat-based model checker for security protocols, business processes, and security apis. In *International Journal on Software Tools for Technology Transfer (18)*, pages 187–204, 2016.

[ACZ13] Alessandro Armando, Roberto Carbone, and Luca Zanetti. Formal Modeling and Automatic Security Analysis of Two-Factor and Two-Channel Authentication Protocols. In *Network and System Security*, LNCS 7873, pages 728–734. Springer, 2013.

[ALFD16] Cain Ashley A., Chiu L., Santiago F., and Still J. D. Swipe authentication: Exploring over-the-shoulder attack performance. In *Advances in Human Factors in Cybersecurity*, pages 327–336. Springer International Publishing, 2016.

[Alt16] Maha Althobaiti. *Assessing usable security of multifactor authentication*. PhD thesis, University of East Anglia, 2016.

[AMV15] O. Almousa, S. Mödersheim, and L. Viganò. *Alice and Bob: Reconciling Formal Models and Implementation*, volume 9465, pages 66–85. 11 2015.

[AMvZE+14] De Luca A., Harbach M., von Zezschwitz E., Maurer M.E., Slawik B. E., Hussmann H., and Smith M. Now you see me, now you don't: Protecting smartphone authentication from shoulder surfers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, page 2937–2946, New York, NY, USA, 2014. Association for Computing Machinery.

[Anda] Android Developers Documentation. Android Guides - Protect against security threats with SafetyNet. `https://developer.android.com/training/safetynet/`.

[Andb] Android Developers Documentation. Android Guides - Security Tips. `https://developer.android.com/training/articles/security-tips#UserData`.

[Andc] Android Developers Documentation. Android Guides - Security with HTTPS and SSL. https://developer.android.com/training/articles/security-ssl.

[APS14] M. Avalle, A. Pironti, and R. Sisto. Formal verification of security protocol implementations: A survey. *Formal Aspects of Computing*, 26, 01 2014.

[AVZ+18] A. Aggarwal, B. Viswanath, L. Zhang, S. Kumar, A. Shah, and P. Kumaraguru. I spy with my little eye: Analysis and detection of spying browser extensions. In *2018 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 47–61, 2018.

[Ayy17] K. Ayyagari. Man in the browser attacks. `https://repository.stcloudstate.edu/cgi/viewcontent.cgi?article=1054&context=msia_etds`, 2017.

[Ban] BankID - electronic identification solution. `https://www.bankid.com/en/`.

[Bla14] Bruno Blanchet. *Automatic Verification of Security Protocols in the Symbolic Model: The Verifier ProVerif*, pages 54–87. Springer International Publishing, 2014.

[BM11] J. Bau and J. C. Mitchell. Security modeling and analysis. *IEEE Security Privacy*, 9(3):18–25, May 2011.

[Bro96] John Brooke. SUS: A quick and dirty usability scale. In *Usability Evaluation in Industry*. Taylor and Francis, 1996.

[BV11] A. Boldyreva and Kumar V. Provable-security analysis of authenticated encryption in kerberos. *IET Information Security*, 5:207–219(12), December 2011.

[CAM11] Neuhaus C., Polze A., and Chowdhuryy M. Survey on healthcare it systems: Standards, regulations and security. 08 2011.

[CC13] Janardan Choubey and Bhaskar Choubey. Secure user authentication in internet banking: A qualitative survey. *International Journal of Innovation, Management and Technology*, 4(2):198–203, 2013.

[CCK15] J. Cho, G. Cho, and H. Kim. Keyboard or keylogger?: A security analysis of third-party keyboards on android. In *2015 13th Annual Conference on Privacy, Security and Trust (PST)*, pages 173–176, 2015.

[CCR16] J. A. Chaudhry, S. A. Chaudhry, and R. Rittenhouse. Phishing attacks and defenses. *International Journal of Security and Its Applications*, 10(1):247–256, 2016.

[CDFN13] Emiliano De Cristofaro, Honglu Du, Julien Freudiger, and Gregory Norcie. Two-Factor or not Two-Factor? A Comparative Usability Study of Two-Factor Authentication. CoRR abs/1309.5344, University College London, 2013.

[Cen16] Centrify. Best Practices for Multi-factor Authentication. `https://www.centrify.com/media/3403844/bpb-best-practices-for-multi-factor-authentication.pdf`, 2016.

[CGDW09] Z. Chen, S. Guo, R. Duan, and S. Wang. Security analysis on mutual authentication against man-in-the-middle attack. In *2009 First International Conference on Information Science and Engineering*, pages 1855–1858, Dec 2009.

[CNB] CNBC. Google is missing out on billions of dollars by not having an app store in China, new data shows. `https://www.cnbc.com/2018/01/17/google-misses-out-on-billions-in-china.html`.

[Com18] European Commission. Market study on telemedicine. `https://ec.europa.eu/health/sites/health/files/ehealth/docs/2018_provision_marketstudy_telemedicine_en.pdf`, 2018.

[DC12] T. Dougan and K. Curran. Man in the browser attacks. *International Journal of Ambient Computing and Intelligence (IJACI)*, 4(1):29–39, 2012.

[DDC18] S. Das, A. Dingman, and L.J. Camp. Why johnny doesn't use two factor a two-phase usability study of the fido u2f security key. In *Financial Cryptography and Data Security*, pages 160–179, Berlin, Heidelberg, 2018. Springer Berlin Heidelberg.

[DDSW10] L. Davi, A. Dmitrienko, A. Sadeghi, and M. Winandy. Privilege escalation attacks on android. volume 6531, pages 346–360, 10 2010.

[DeF11] Dimitri DeFigueiredo. The Case for Mobile Two-Factor Authentication. *IEEE Security and Privacy*, 9:81–85, 2011.

[DLRS14] A. Dmitrienko, C. Liebchen, C. Rossow, and A. Sadeghi. Security Analysis of Mobile Two-Factor Authentication Schemes. *Intel Technology Journal*, 18(4):138–161, 2014.

[DMC15] Q. Do, B. Martini, and K. R. Choo. Exfiltrating data from android devices. *Computers & Security*, 48:74 – 91, 2015.

[DY81] D. Dolev and A. C. Yao. On the security of public key protocols. In *Proceedings of the 22nd Annual Symposium on Foundations of Computer Science*, SFCS '81, page 350–357, USA, 1981. IEEE Computer Society.

[EU19] Brumaghin E. and H. Unterbrink. New hawkeye reborn variant emerges following ownership change. https://blog.talosintelligence.com/2019/04/hawkeye-reborn.html, 2019.

[Eur13a] European Banking Authority. Recommendations for the Security of Internet Payments. https://www.ecb.europa.eu/pub/pdf/other/recommendationssecurityinternetpaymentsoutcomeofpcfinalversionafterpc20130pdf, 2013.

[Eur13b] European Banking Authority. Recommendations for the Security of Mobile Payments - DRAFT. https://www.ecb.europa.eu/paym/cons/pdf/131120/recommendationsforthesecurityofmobilepaymentsdraftpc201311en.pdf, 2013.

[Eur14] European Parliament and Council. Regulation (EU) No 910/2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC. https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32014R0910, 2014.

[Eur15] European Banking Authority. Directive 2015/2366 of the European Parliament and of the Council on payment services in the internal market (PSD2). https://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX:32015L2366, 2015.

[Eur17] European Banking Authority. Regulatory Technical Standards on Strong Customer Authentication and common and secure communication under Article 98 of PSD2. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32018R0389&from=EN, 2017.

[Eur18] Eurostat. Internet banking on the rise. https://ec.europa.eu/eurostat/web/products-eurostat-news/-/DDN-20180115-1, 2018.

[Gar16] Keith Garrett. Vulnerabililty analysis of multi-factor authentication protocols. 2016.

[Gem15] Gemalto. Authentication BestPractices: Put control where it belongs. http://www2.gemalto.com/email/2011/authsomethingstronger/whitepaper/Authentication_Best_Practices_WP(EN)_A4_web.pdf, 2015.

[Goo17] Google Developers. https://developers.google.com/china/, 2017.

[GPP+16] D. Genkin, L. Pachmanov, I. Pipman, E. Tromer, and Y. Yarom. Ecdsa key extraction from mobile devices via nonintrusive physical side channels. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1626–1638, New York, NY, USA, 2016. Association for Computing Machinery.

[Güh06] Philipp Gühring. Concepts against man-in-the-browser attacks, 2006.

[Had10] C. Hadnagy. *Social engineering: The art of human hacking*. John Wiley & Sons, 2010.

[Hag07] A. M. Hagalisletto. Analyzing two-factor authentication devices. Technical report, University of Oslo, 2007.

[HM18] Vincent Haupert and Tilo Müller. On App-based Matrix Code Authentication in Online Banking. In *Proceedings of International Conference on Information Systems Security and Privacy*, pages 149–160. SciTePress, 2018.

[HMN15] T. Halevi, N. Memon, and O. Nov. Spear-phishing in the wild: A real-world study of personality, phishing self-efficacy and vulnerability to spear-phishing attacks. *SSRN Electronic Journal*, 01 2015.

[ISO18] ISO. ISO 9241-11:2018(en), Ergonomics of human-system interaction – Part 11: Usability: Definitions and concepts. 2018.

[JCL+17] Q. Jiang, Z. Chen, B. Li, J. Shen, L. Yang, and J. Ma. Security analysis and improvement of bio-hashing based three-factor authentication scheme for telecare medical information systems. *Journal of Ambient Intelligence and Humanized Computing*, 9, 2017.

[JFR17] Aviv Adam J., Davin J. T.and Wolf F., and Kuber R. Towards baselines for shoulder surfing on mobile authentication. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, ACSAC 2017, page 486–498, New York, NY, USA, 2017. Association for Computing Machinery.

[JJIL16] Y. Jo, S. Jeon, J. Im, and M. Lee. Security analysis and improvement of fingerprint authentication for smartphones. *Mobile Information Systems*, 2016:1–11, 03 2016.

[JK18] C. Jacomme and S. Kremer. An extensive formal analysis of multi-factor authentication protocols. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 1–15, July 2018.

[Kas13] Kaspersky Lab. Online and Mobile Banking Threats. `https://media.kaspersky.com/en/business-security/online-and-mobile-banking-threats-kaspersky-whitepaper.pdf?icid=en-UK:ent-content`, 2013.

[Kas19] Kaspersky. Spam and phishing in q2 2019. `https://securelist.com/spam-and-phishing-in-q2-2019/92379/`, 2019.

[KHHW15] K. Krombholz, H. Hobel, M. Huber, and E. Weippl. Advanced social engineering attacks. *Journal of Information Security and Applications*, 22:113 – 122, 2015. Special Issue on Security of Information and Networks.

[KPCS15] Kat Krol, Eleni Philippou, Emiliano De Cristofaro, and Martina Angela Sasse. "They brought in the horrible key ring thing!" Analysing the Usability of Two-Factor Authentication in UK Online Banking. CoRR abs/1501.04434, University College London, 2015.

[LB19] Bošnjak L. and Brumen B. Shoulder surfing: From an experimental study to a comparative framework. *International Journal of Human-Computer Studies*, 130:1 – 20, 2019.

[LM11] J. Long and K.D. Mitnick. *No Tech Hacking: A Guide to Social Engineering, Dumpster Diving, and Shoulder Surfing*. Elsevier Science, 2011.

[Low96] G. Lowe. Breaking and fixing the Needham-Schroeder Public-Key Protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 147–166. Springer, 1996.

[Low97] G. Lowe. A hierarchy of authentication specifications. In *Proceedings 10th Computer Security Foundations Workshop*, pages 31–43. IEEE, 1997.

[MA15] A. Memon and A. Anwar. Colluding apps: Tomorrow's mobile malware threat. *IEEE Security & Privacy*, 13:77–81, 11 2015.

[Mae16] E. Maelstrom. Olympic vision keylogger and bec scams. `https://info.phishlabs.com/blog/olympic-vision-keylogger-and-bec-scams`, 2016.

[MBSS13] Collin Mulliner, Ravishankar Borgaonkar, Patrick Stewin, and Jean-Pierre Seifert. Sms-based one-time passwords: Attacks and defense. In Konrad Rieck,

Patrick Stewin, and Jean-Pierre Seifert, editors, *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 150–159, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[MKTM16] Silvere Mavoungou, Georges Kaddoum, Mostafa Taha, and Georges Matar. Survey on threats and attacks on mobile networks. *IEEE Access*, 4:4543–4572, 2016.

[MMvZEF17] Eiband M., Khamis M., von Zezschwitz E., and Hussmann H. Alt F. Understanding shoulder surfing in the wild: Stories from users and observers. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 4254–4265, New York, NY, USA, 2017. Association for Computing Machinery.

[MS13] F. Mohsen and M. Shehab. Android keylogging threat. In *9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 545–552, Oct 2013.

[MSCB13] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The tamarin prover for the symbolic analysis of security protocols. In *International Conference on Computer Aided Verification*, pages 696–701. Springer, 2013.

[NIS17] NIST. Special Publication - Digital Identity Guidelines. `https://pages.nist.gov/800-63-3/`, 2017.

[OBM+18] A. Ometov, S. Bezzateev, N. Mäkitalo, S. Andreev, T. Mikkonen, and Y. Koucheryavy. Multi-factor authentication: A survey. *Cryptography*, 2(1):1, 2018.

[PCI16a] PCI Security Standards Council. Data Security Standard. `https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2.pdf`, 2016.

[PCI16b] PCI Security Standards Council. Payment Application Data Security Standard. `https://www.pcisecuritystandards.org/documents/PA-DSS_v3-2.pdf`, 2016.

[PCI17] PCI Security Standards Council. Information Supplement - Multi-Factor Authentication. `https://www.pcisecuritystandards.org/pdfs/Multi-Factor-Authentication-Guidance-v1.pdf`, 2017.

[Pin09] PingIdentity. Multi-Factor Authentication: Best Practices for Securing the Modern Digital Enterprise. `https://www.pingidentity.com/content/dam/ping-6-2-assets/Assets/white-papers/en/mfa-best-practices-securing-modern-digital-enterprise-3001.pdf?id=b6322a80-f285-11e3-ac10-0800200c9a66`, 2009.

[Pro08] AVANTSSAR Project. Deliverable D2.3 (update) ASLan++ specification and tutorial. http://www.avantssar.eu/pdf/deliverables/avantssar-d2-3_update.pdf. Also available as https://sites.google.com/fbk.eu/mobile-mfa-formal-analysis, 2008.

[PRW17] Olivier Pereira, Florentin Rochet, and Cyrille Wiedling. Formal analysis of the fido 1. x protocol. In *International Symposium on Foundations and Practice of Security*, pages 68–82. Springer, 2017.

[R.19] Keegan R. Technical advisory: Private key extraction from qualcomm hardware-backed keystores. https://www.nccgroup.trust/us/our-research/private-key-extraction-qualcomm-keystore/, 2019.

[S. 16] S. Kiljan, K. Simoens, D. De Cock, M. Van Eekelen, H. Vranken. A Survey of Authentication and Communications Security in Online Banking. *ACM Computer Surveys*, 49(4):61:1–61:35, 2016.

[SCRV] Giada Sciarretta, Roberto Carbone, Silvio Ranise, and Luca Viganò. Formal analysis of mobile multi-factor authentication with single sign-on login. *ACM Trans. Inf. Syst. Secur.*, 0(ja).

[SCRV18] Giada Sciarretta, Roberto Carbone, Silvio Ranise, and Luca Viganò. Design, formal specification and analysis of multi-factor authentication solutions with a single sign-on experience. In Lujo Bauer and Ralf Küsters, editors, *Principles of Security and Trust*, pages 188–213. Springer International Publishing, 2018.

[Shi19] T. Shishkova. Riltok mobile trojan: A banker with global reach. https://securelist.com/mobile-banker-riltok/91374/, 2019.

[ST16] M. Sabt and J. Traoré. Breaking into the keystore: A practical forgery attack against android keystore. volume 9879, pages 531–548, 09 2016.

[TFT+16] T. Tsuchiya, M. Fujita, K. Takahashi, T. Kato, F. Magata, Y. Teshigawara, R. Sasaki, and M. Nishigaki. Secure communication protocol between a human and a bank server for preventing man-in-the-browser attacks. In T. Tryfonas, editor, *Human Aspects of Information Security, Privacy, and Trust*, pages 77–88, Cham, 2016. Springer International Publishing.

[Ver17] Verizon. 2017 data breach investigations report. https://enterprise.verizon.com/resources/reports/2017_dbir.pdf, 2017.

[Vig06] Luca Viganò. Automated security protocol analysis with the avispa tool. *Electronic Notes in Theoretical Computer Science*, 155:61–86, 2006.

[Vir14] Virus Bulletin. The Evolution of Webinjects. https://www.virusbulletin.com/uploads/pdf/conference/vb2014/VB2014-Boutin.pdf, 2014.

[vOM12] David von Oheimb and Sebastian Mödersheim. Aslan++ — a formal security specification language for distributed systems. In Bernhard K. Aichernig, Frank S. de Boer, and Marcello M. Bonsangue, editors, *Formal Methods for Components and Objects*, pages 1–22. Springer Berlin Heidelberg, 2012.

[WDCJ09] Catherine S. Weir, Gary Douglas, Martin Carruthers, and Mervyn Jack. User perceptions of security, convenience and usability for ebanking authentication tokens. *Computers & Security*, 28(1):47–62, 2009.

[WDRJ10] Catherine S. Weir, Gary Douglas, Tim Richardson, and Mervyn Jack. Usable security: User preferences for authentication methods in eBanking and the effects of experience. *Interacting with Computers*, 22(3):153–164, 2010.

[XPW+14] L. Xing, X. Pan, R. Wang, K. Yuan, and X. Wang. Upgrading your android, elevating my malware: Privilege escalation through mobile os updating. In *2014 IEEE Symposium on Security and Privacy*, pages 393–408, 2014.

Appendix

Example of questionnaire proposed by MuFASA

Here we provide an example of how the user fills the questionnaire to obtain Nordea1 protocol (3.2). Notice that the sequence of the reported questions only represent a specific path in the interpretation tree.

Table 1: Questionnaire proposed by MuFASA for modeling Nordea1.

1. From which Endpoint do you start the protocol?
   - ◉ From a desktop computer
   - ○ From a smartphone
2. What is your 1st operation?
   - ◉ I insert some secret credentials (e.g., a password on a website)
   - ○ I read a value from a list
   - ○ I use a device (e.g., a card reader)
   - ○ I use a software (e.g., an app on my smartphone)
   - ○ I send/receive something on my mobile phone (e.g., an SMS)
   - ○ None, I am authenticated

3. What is your 2nd operation?

- ○ I insert some secret credentials (e.g., a password on a website)
- ○ I read a value from a list
- ⊘ I use a device (e.g., a card reader)
- ○ I use a software (e.g., an app on my smartphone)
- ○ I send/receive something on my mobile phone (e.g., an SMS)
- ○ None, I am authenticated

3a. Is your device connected to the endpoint?

- ⊘ No, it is isolated
- ○ Yes (e.g., through a USB cable)

3b. Does it read some sort of input code?

- ○ No
- ○ Yes, it scans an optic code (e.g., barcode or QR code)
- ○ Yes, it automatically sent by the endpoint
- ⊘ Yes, I personally digit it (e.g., a code displayed on a website)

3c. Among the followings, what do you need to use the device?

- ○ Nothing
- ⊘ I must insert a secret code/pin
- ○ I must scan a part of my body (e.g., my fingerprint)

3d. Does it recap the ongoing operation and ask for your confirmation?

- ⊘ No
- ○ Yes (e.g., "Your are paying *x*$ to *y*. Confirm?")

3e. Does it return some code that you have to copy somewhere?

- ○ No
- ⊘ Yes

4. What is your 3rd operation?

- ○ I insert some secret credentials (e.g., a password on a website)
- ○ I read a value from a list
- ○ I use a device (e.g., a card reader)
- ○ I use a software (e.g., an app on my smartphone)
- ○ I send/receive something on my mobile phone (e.g., an SMS)
- ⊘ None, I am authenticated

MuFASA Forms

In this section, we report all the outcomes of the possible form sequences in MuFASA.

Table 2: Table of possible sequences of forms after selecting "Device" option; †since the device is not connected to the endpoint, there are no other possibilities than manual copy; ◇since the authenticator is not on the same platform of the endpoint, it cannot receive a "direct input"; *since the authenticator does not receive an input, it cannot show info on the ongoing operation.

FE	**FA**	**FC** Endp. Connect.	**F1** Input	**F2** Addit. Factor	**F3** Confirm.	**F4** Manual copy	**Authenticator**
Any	A3 (Device)	1 (No)	1 (No)	1 (Nog)	1 (No)	~~1 (No)~~	not possible†
Any	A3 (Device)	1 (No)	1 (No)	1 (No)	1 (No)	2 (Yes)	🖩[O] $\gg_h$ `otp`
Any	A3 (Device)	1 (No)	1 (No)	1 (No)	~~2 (Yes)~~	–	not possible*
Any	A3 (Device)	1 (No)	1 (No)	2 (PIN)	1 (No)	~~1 (No)~~	not possible†
Any	A3 (Device)	1 (No)	1 (No)	2 (PIN)	1 (No)	2 (Yes)	🖩[O,K] $\gg_h$ `otp`
Any	A3 (Device)	1 (No)	1 (No)	2 (PIN)	~~2 (Yes)~~	–	not possible*
Any	A3 (Device)	1 (No)	1 (No)	3 (Biom.)	1 (No)	~~1 (No)~~	not possible†
Any	A3 (Device)	1 (No)	1 (No)	3 (Biom.)	1 (No)	2 (Yes)	🖩[O,I] $\gg_h$ `otp`
Any	A3 (Device)	1 (No)	1 (No)	3 (Biom.)	~~2 (Yes)~~	–	not possible*
Any	A3 (Device)	1 (No)	2 (Scan)	1 (No)	1 (No)	~~1 (No)~~	not possible†
Any	A3 (Device)	1 (No)	2 (Scan)	1 (No)	1 (No)	2 (Yes)	`opid`$\gg_o$ 🖩[O] $\gg_h$ `otp`
Any	A3 (Device)	1 (No)	2 (Scan)	1 (No)	2 (Yes)	~~1 (No)~~	not possible†
Any	A3 (Device)	1 (No)	2 (Scan)	1 (No)	2 (Yes)	2 (Yes)	`opid`$\gg_o$ 🖩?[O] $\gg_h$ `otp`
Any	A3 (Device)	1 (No)	2 (Scan)	2 (PIN)	1 (No)	~~1 (No)~~	not possible†
Any	A3 (Device)	1 (No)	2 (Scan)	2 (PIN)	1 (No)	2 (Yes)	`opid`$\gg_o$ 🖩[O,K] $\gg_h$ `otp`
Any	A3 (Device)	1 (No)	2 (Scan)	2 (PIN)	2 (Yes)	~~1 (No)~~	not possible†
Any	A3 (Device)	1 (No)	2 (Scan)	2 (PIN)	2 (Yes)	2 (Yes)	`opid`$\gg_o$ 🖩?[O,K] $\gg_h$ `otp`
Any	A3 (Device)	1 (No)	2 (Scan)	3 (Biom.)	1 (No)	~~1 (No)~~	not possible†
Any	A3 (Device)	1 (No)	2 (Scan)	3 (Biom.)	1 (No)	2 (Yes)	`opid`$\gg_o$ 🖩[O,I] $\gg_h$ `otp`
Any	A3 (Device)	1 (No)	2 (Scan)	3 (Biom.)	2 (Yes)	~~1 (No)~~	not possible†
Any	A3 (Device)	1 (No)	2 (Scan)	3 (Biom.)	2 (Yes)	2 (Yes)	`opid`$\gg_o$ 🖩?[O,I] $\gg_h$ `otp`
Any	A3 (Device)	1 (No)	~~3 (Direct)~~	–	–	–	not possible◇
Any	A3 (Device)	1 (No)	4 (Manual)	1 (No)	1 (No)	~~1 (No)~~	not possible†
Any	A3 (Device)	1 (No)	4 (Manual)	1 (No)	1 (No)	2 (Yes)	`opid`$\gg_h$ 🖩[O] $\gg_h$ `otp`
Any	A3 (Device)	1 (No)	4 (Manual)	1 (No)	2 (Yes)	~~1 (No)~~	not possible†
Any	A3 (Device)	1 (No)	4 (Manual)	1 (No)	2 (Yes)	2 (Yes)	`opid`$\gg_h$ 🖩?[O] $\gg_h$ `otp`
Any	A3 (Device)	1 (No)	4 (Manual)	2 (PIN)	1 (No)	~~1 (No)~~	not possible†
Any	A3 (Device)	1 (No)	4 (Manual)	2 (PIN)	1 (No)	2 (Yes)	`opid`$\gg_h$ 🖩[O,K] $\gg_h$ `otp`
Any	A3 (Device)	1 (No)	4 (Manual)	2 (PIN)	2 (Yes)	~~1 (No)~~	not possible†

Any	A3 (Device)	1 (No)	4 (Manual)	2 (PIN)	2 (Yes)	2 (Yes)	opid$\gg_h$ 🖩?[O,K] $\gg_h$ otp
Any	A3 (Device)	1 (No)	4 (Manual)	3 (Biom.)	1 (No)	~~1 (No)~~	not possible[†]
Any	A3 (Device)	1 (No)	4 (Manual)	3 (Biom.)	1 (No)	2 (Yes)	opid$\gg_h$ 🖩[O,K] $\gg_h$ otp
Any	A3 (Device)	1 (No)	4 (Manual)	3 (Biom.)	2 (Yes)	~~1 (No)~~	not possible[†]
Any	A3 (Device)	1 (No)	4 (Manual)	3 (Biom.)	2 (Yes)	2 (Yes)	opid$\gg_h$ 🖩?[O,I] $\gg_h$ otp
Any	A3 (Device)	2 (Yes)	1 (No)	1 (No)	1 (No)	1 (No)	🖩[O] $\gg_i$ otp
Any	A3 (Device)	2 (Yes)	1 (No)	1 (No)	1 (No)	2 (Yes)	🖩[O] $\gg_h$ otp
Any	A3 (Device)	2 (Yes)	1 (No)	1 (No)	~~2 (Yes)~~	–	not possible[*]
Any	A3 (Device)	2 (Yes)	1 (No)	2 (PIN)	1 (No)	1 (No)	🖩[O,K] $\gg_i$ otp
Any	A3 (Device)	2 (Yes)	1 (No)	2 (PIN)	1 (No)	2 (Yes)	🖩[O,K] $\gg_h$ otp
Any	A3 (Device)	2 (Yes)	1 (No)	2 (PIN)	~~2 (Yes)~~	–	not possible[*]
Any	A3 (Device)	2 (Yes)	1 (No)	3 (Biom.)	1 (No)	1 (No)	🖩[O,I] $\gg_i$ otp
Any	A3 (Device)	2 (Yes)	1 (No)	3 (Biom.)	1 (No)	2 (Yes)	🖩[O,I] $\gg_h$ otp
Any	A3 (Device)	2 (Yes)	1 (No)	3 (Biom.)	~~2 (Yes)~~	–	not possible[*]
Any	A3 (Device)	2 (Yes)	2 (Scan)	1 (No)	1 (No)	1 (No)	opid$\gg_o$ 🖩[O] $\gg_i$ otp
Any	A3 (Device)	2 (Yes)	2 (Scan)	1 (No)	1 (No)	2 (Yes)	opid$\gg_o$ 🖩[O] $\gg_h$ otp
Any	A3 (Device)	2 (Yes)	2 (Scan)	1 (No)	2 (Yes)	2 (No)	opid$\gg_o$ 🖩?[O] $\gg_i$ otp
Any	A3 (Device)	2 (Yes)	2 (Scan)	1 (No)	2 (Yes)	2 (Yes)	opid$\gg_o$ 🖩?[O] $\gg_h$ otp
Any	A3 (Device)	2 (Yes)	2 (Scan)	2 (PIN)	1 (No)	1 (No)	opid$\gg_o$ 🖩[O,K] $\gg_i$ otp
Any	A3 (Device)	2 (Yes)	2 (Scan)	2 (PIN)	1 (No)	2 (Yes)	opid$\gg_o$ 🖩[O,K] $\gg_h$ otp
Any	A3 (Device)	2 (Yes)	2 (Scan)	2 (PIN)	2 (Yes)	1 (No)	opid$\gg_o$ 🖩?[O,K] $\gg_i$ otp
Any	A3 (Device)	2 (Yes)	2 (Scan)	2 (PIN)	2 (Yes)	2 (Yes)	opid$\gg_o$ 🖩?[O,K] $\gg_h$ otp
Any	A3 (Device)	2 (Yes)	2 (Scan)	3 (Biom.)	1 (No)	1 (No)	opid$\gg_o$ 🖩[O,I] $\gg_i$ otp
Any	A3 (Device)	2 (Yes)	2 (Scan)	3 (Biom.)	1 (No)	2 (Yes)	opid$\gg_o$ 🖩[O,I] $\gg_h$ otp
Any	A3 (Device)	2 (Yes)	2 (Scan)	3 (Biom.)	2 (Yes)	1 (No)	opid$\gg_o$ 🖩?[O,I] $\gg_i$ otp
Any	A3 (Device)	2 (Yes)	2 (Scan)	3 (Biom.)	2 (Yes)	2 (Yes)	opid$\gg_o$ 🖩?[O,I] $\gg_h$ otp
Any	A3 (Device)	2 (Yes)	3 (Direct)	1 (No)	1 (No)	1 (No)	opid$\gg_i$ 🖩[O] $\gg_i$ otp
Any	A3 (Device)	2 (Yes)	3 (Direct)	1 (No)	1 (No)	2 (Yes)	opid$\gg_i$ 🖩[O] $\gg_h$ otp
Any	A3 (Device)	2 (Yes)	3 (Direct)	1 (No)	2 (Yes)	2 (No)	opid$\gg_i$ 🖩?[O] $\gg_i$ otp
Any	A3 (Device)	2 (Yes)	3 (Direct)	1 (No)	2 (Yes)	2 (Yes)	opid$\gg_i$ 🖩?[O] $\gg_h$ otp
Any	A3 (Device)	2 (Yes)	3 (Direct)	2 (PIN)	1 (No)	1 (No)	opid$\gg_i$ 🖩[O,K] $\gg_i$ otp
Any	A3 (Device)	2 (Yes)	3 (Direct)	2 (PIN)	1 (No)	2 (Yes)	opid$\gg_i$ 🖩[O,K] $\gg_h$ otp
Any	A3 (Device)	2 (Yes)	3 (Direct)	2 (PIN)	2 (Yes)	1 (No)	opid$\gg_i$ 🖩?[O,K] $\gg_i$ otp
Any	A3 (Device)	2 (Yes)	3 (Direct)	2 (PIN)	2 (Yes)	2 (Yes)	opid$\gg_i$ 🖩?[O,K] $\gg_h$ otp
Any	A3 (Device)	2 (Yes)	3 (Direct)	3 (Biom.)	1 (No)	1 (No)	opid$\gg_i$ 🖩[O,I] $\gg_i$ otp
Any	A3 (Device)	2 (Yes)	3 (Direct)	3 (Biom.)	1 (No)	2 (Yes)	opid$\gg_i$ 🖩[O,I] $\gg_h$ otp
Any	A3 (Device)	2 (Yes)	3 (Direct)	3 (Biom.)	2 (Yes)	1 (No)	opid$\gg_i$ 🖩?[O,I] $\gg_i$ otp
Any	A3 (Device)	2 (Yes)	3 (Direct)	3 (Biom.)	2 (Yes)	2 (Yes)	opid$\gg_i$ 🖩?[O,I] $\gg_h$ otp
Any	A3 (Device)	2 (Yes)	4 (Manual)	1 (No)	1 (No)	1 (No)	opid$\gg_h$ 🖩[O] $\gg_i$ otp
Any	A3 (Device)	2 (Yes)	4 (Manual)	1 (No)	1 (No)	2 (Yes)	opid$\gg_h$ 🖩[O] $\gg_h$ otp
Any	A3 (Device)	2 (Yes)	4 (Manual)	1 (No)	2 (Yes)	2 (No)	opid$\gg_h$ 🖩?[O] $\gg_i$ otp

Any	A3 (Device)	2 (Yes)	4 (Manual)	1 (No)	2 (Yes)	2 (Yes)	opid»$_h$ 🖩?[O] »$_h$ otp
Any	A3 (Device)	2 (Yes)	4 (Manual)	2 (PIN)	1 (No)	1 (No)	opid»$_h$ 🖩[O,K] »$_i$ otp
Any	A3 (Device)	2 (Yes)	4 (Manual)	2 (PIN)	1 (No)	2 (Yes)	opid»$_h$ 🖩[O,K] »$_h$ otp
Any	A3 (Device)	2 (Yes)	4 (Manual)	2 (PIN)	2 (Yes)	1 (No)	opid»$_h$ 🖩?[O,K] »$_i$ otp
Any	A3 (Device)	2 (Yes)	4 (Manual)	2 (PIN)	2 (Yes)	2 (Yes)	opid»$_h$ 🖩?[O,K] »$_h$ otp
Any	A3 (Device)	2 (Yes)	4 (Manual)	3 (Biom.)	1 (No)	1 (No)	opid»$_h$ 🖩[O,I] »$_i$ otp
Any	A3 (Device)	2 (Yes)	4 (Manual)	3 (Biom.)	1 (No)	2 (Yes)	opid»$_h$ 🖩[O,K] »$_h$ otp
Any	A3 (Device)	2 (Yes)	4 (Manual)	3 (Biom.)	2 (Yes)	1 (No)	opid»$_h$ 🖩?[O,I] »$_i$ otp
Any	A3 (Device)	2 (Yes)	4 (Manual)	3 (Biom.)	2 (Yes)	2 (Yes)	opid»$_h$ 🖩?[O,I] »$_h$ otp

Table 3: Table of possible sequences of forms after selecting "Software" option; $^{\diamond}$since the authenticator is not on the same platform of the endpoint, it cannot receive a "direct input"; *since the authenticator does not receive an input, it cannot show info on the ongoing operation.

E	**FA**	**FI** Installed on	**F1** Input	**F2** Addit. Factor	**F3** Confirm.	**F4** Manual copy	**Authenticator**
Brows.	A4 (Softw.)	1 (Desktop)	1 (No)	1 (No)	1 (No)	1 (No)	🔐[O] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	1 (No)	1 (No)	1 (No)	2 (Yes)	🔐[O] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	1 (No)	1 (No)	~~2 (Yes)~~	–	not possible*
Brows.	A4 (Softw.)	1 (Desktop)	1 (No)	2 (PIN)	1 (No)	1 (No)	🔐[O,K] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	1 (No)	2 (PIN)	1 (No)	2 (Yes)	🔐[O,K] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	1 (No)	2 (PIN)	~~2 (Yes)~~	–	not possible*
Brows.	A4 (Softw.)	1 (Desktop)	1 (No)	3 (Biom.)	1 (No)	1 (No)	🔐[O,I] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	1 (No)	3 (Biom.)	1 (No)	2 (Yes)	🔐[O,I] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	1 (No)	3 (Biom.)	~~2 (Yes)~~	–	not possible*
Brows.	A4 (Softw.)	1 (Desktop)	2 (scan)	1 (No)	1 (No)	1 (No)	opid»$_o$ 🔐[O] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	2 (scan)	1 (No)	1 (No)	2 (Yes)	opid»$_o$ 🔐[O] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	2 (scan)	1 (No)	2 (Yes)	1 (No)	opid»$_o$ 🔐?[O] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	2 (scan)	1 (No)	2 (Yes)	2 (Yes)	opid»$_o$ 🔐?[O] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	2 (scan)	2 (PIN)	1 (No)	1 (No)	opid»$_o$ 🔐[O,K] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	2 (scan)	2 (PIN)	1 (No)	2 (Yes)	opid»$_o$ 🔐[O,K] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	2 (scan)	2 (PIN)	2 (Yes)	1 (No)	opid»$_o$ 🔐?[O,K] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	2 (scan)	2 (PIN)	2 (Yes)	2 (Yes)	opid»$_o$ 🔐?[O,K] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	2 (scan)	3 (Biom.)	1 (No)	1 (No)	opid»$_o$ 🔐[O,I] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	2 (scan)	3 (Biom.)	1 (No)	2 (Yes)	opid»$_o$ 🔐[O,I] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	2 (scan)	3 (Biom.)	2 (Yes)	1 (No)	opid»$_o$ 🔐?[O,I] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	2 (scan)	3 (Biom.)	2 (Yes)	2 (Yes)	opid»$_o$ 🔐?[O,I] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	3 (direct)	1 (No)	1 (No)	1 (No)	opid»$_i$ 🔐[O] »$_i$ otp

Brows.	A4 (Softw.)	1 (Desktop)	3 (direct)	1 (No)	1 (No)	2 (Yes)	opid»$_i$ [O] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	3 (direct)	1 (No)	2 (Yes)	1 (No)	opid»$_i$ [O] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	3 (direct)	1 (No)	2 (Yes)	2 (Yes)	opid»$_i$ [O] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	3 (direct)	2 (PIN)	1 (No)	1 (No)	opid»$_i$ [O,K] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	3 (direct)	2 (PIN)	1 (No)	2 (Yes)	opid»$_i$ [O,K] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	3 (direct)	2 (PIN)	2 (Yes)	1 (No)	opid»$_i$ [O,K] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	3 (direct)	2 (PIN)	2 (Yes)	2 (Yes)	opid»$_i$ [O,K] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	3 (direct)	3 (Biom.)	1 (No)	1 (No)	opid»$_i$ [O,I] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	3 (direct)	3 (Biom.)	1 (No)	2 (Yes)	opid»$_i$ [O,I] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	3 (direct)	3 (Biom.)	2 (Yes)	1 (No)	opid»$_i$ [O,I] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	3 (direct)	3 (Biom.)	2 (Yes)	2 (Yes)	opid»$_i$ [O,I] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	4 (manual)	1 (No)	1 (No)	1 (No)	opid»$_h$ [O] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	4 (manual)	1 (No)	1 (No)	2 (Yes)	opid»$_h$ [O] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	4 (manual)	1 (No)	2 (Yes)	2 (No)	opid»$_h$ [O] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	4 (manual)	1 (No)	2 (Yes)	2 (Yes)	opid»$_h$ [O] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	4 (manual)	2 (PIN)	1 (No)	1 (No)	opid»$_h$ [O,K] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	4 (manual)	2 (PIN)	1 (No)	2 (Yes)	opid»$_h$ [O,K] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	4 (manual)	2 (PIN)	2 (Yes)	1 (No)	opid»$_h$ [O,K] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	4 (manual)	2 (PIN)	2 (Yes)	2 (Yes)	opid»$_h$ [O,K] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	4 (manual)	3 (Biom.)	1 (No)	1 (No)	opid»$_h$ [O,I] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	4 (manual)	3 (Biom.)	1 (No)	2 (Yes)	opid»$_h$ [O,K] »$_h$ otp
Brows.	A4 (Softw.)	1 (Desktop)	4 (manual)	3 (Biom.)	2 (Yes)	1 (No)	opid»$_h$ [O,I] »$_i$ otp
Brows.	A4 (Softw.)	1 (Desktop)	4 (manual)	3 (Biom.)	2 (Yes)	2 (Yes)	opid»$_h$ [O,I] »$_h$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	1 (No)	1 (No)	1 (No)	1 (No)	[O] »$_n$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	1 (No)	1 (No)	1 (No)	2 (Yes)	[O] »$_h$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	1 (No)	1 (No)	~~2 (Yes)~~	–	not possible*
Brows.	A4 (Softw.)	2 (Smartph.)	1 (No)	2 (PIN)	1 (No)	1 (No)	[O,K] »$_n$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	1 (No)	2 (PIN)	1 (No)	2 (Yes)	[O,K] »$_h$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	1 (No)	2 (PIN)	~~2 (Yes)~~	–	not possible*
Brows.	A4 (Softw.)	2 (Smartph.)	1 (No)	3 (Biom.)	1 (No)	1 (No)	[O,I] »$_n$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	1 (No)	3 (Biom.)	1 (No)	2 (Yes)	[O,I] »$_h$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	1 (No)	3 (Biom.)	~~2 (Yes)~~	–	not possible*
Brows.	A4 (Softw.)	2 (Smartph.)	2 (scan)	1 (No)	1 (No)	1 (No)	opid»$_o$ [O] »$_n$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	2 (scan)	1 (No)	1 (No)	2 (Yes)	opid»$_o$ [O] »$_h$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	2 (scan)	1 (No)	2 (Yes)	2 (No)	opid»$_o$ [O] »$_n$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	2 (scan)	1 (No)	2 (Yes)	2 (Yes)	opid»$_o$ [O] »$_h$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	2 (scan)	2 (PIN)	1 (No)	1 (No)	opid»$_o$ [O,K] »$_n$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	2 (scan)	2 (PIN)	1 (No)	2 (Yes)	opid»$_o$ [O,K] »$_h$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	2 (scan)	2 (PIN)	2 (Yes)	1 (No)	opid»$_o$ [O,K] »$_n$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	2 (scan)	2 (PIN)	2 (Yes)	2 (Yes)	opid»$_o$ [O,K] »$_h$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	2 (scan)	3 (Biom.)	1 (No)	1 (No)	opid»$_o$ [O,I] »$_n$ otp

Brows.	A4 (Softw.)	2 (Smartph.)	2 (scan)	3 (Biom.)	1 (No)	2 (Yes)	opid»$_o$ [O,I] »$_h$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	2 (scan)	3 (Biom.)	2 (Yes)	1 (No)	opid»$_o$ [O,I] »$_n$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	2 (scan)	3 (Biom.)	2 (Yes)	2 (Yes)	opid»$_o$ [O,I] »$_h$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	~~3 (direct)~~	–	–	–	not possible
Brows.	A4 (Softw.)	2 (Smartph.)	4 (manual)	1 (No)	1 (No)	1 (No)	opid»$_h$ [O] »$_n$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	4 (manual)	1 (No)	1 (No)	2 (Yes)	opid»$_h$ [O] »$_h$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	4 (manual)	1 (No)	2 (Yes)	2 (No)	opid»$_h$ [O] »$_n$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	4 (manual)	1 (No)	2 (Yes)	2 (Yes)	opid»$_h$ [O] »$_h$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	4 (manual)	2 (PIN)	1 (No)	1 (No)	opid»$_h$ [O,K] »$_n$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	4 (manual)	2 (PIN)	1 (No)	2 (Yes)	opid»$_h$ [O,K] »$_h$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	4 (manual)	2 (PIN)	2 (Yes)	1 (No)	opid»$_h$ [O,K] »$_n$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	4 (manual)	2 (PIN)	2 (Yes)	2 (Yes)	opid»$_h$ [O,K] »$_h$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	4 (manual)	3 (Biom.)	1 (No)	1 (No)	opid»$_h$ [O,I] »$_n$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	4 (manual)	3 (Biom.)	1 (No)	2 (Yes)	opid»$_h$ [O,I] »$_h$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	4 (manual)	3 (Biom.)	2 (Yes)	1 (No)	opid»$_h$ [O,I] »$_n$ otp
Brows.	A4 (Softw.)	2 (Smartph.)	4 (manual)	3 (Biom.)	2 (Yes)	2 (Yes)	opid»$_h$ [O,I] »$_h$ otp
App	A4 (Softw.)	1 (Desktop)	1 (No)	1 (No)	1 (No)	1 (No)	[O] »$_n$ otp
App	A4 (Softw.)	1 (Desktop)	1 (No)	1 (No)	1 (No)	2 (Yes)	[O] »$_h$ otp
App	A4 (Softw.)	1 (Desktop)	1 (No)	1 (No)	~~2 (Yes)~~	–	not possible*
App	A4 (Softw.)	1 (Desktop)	1 (No)	2 (PIN)	1 (No)	1 (No)	[O,K] »$_n$ otp
App	A4 (Softw.)	1 (Desktop)	1 (No)	2 (PIN)	1 (No)	2 (Yes)	[O,K] »$_h$ otp
App	A4 (Softw.)	1 (Desktop)	1 (No)	2 (PIN)	~~2 (Yes)~~	–	not possible*
App	A4 (Softw.)	1 (Desktop)	1 (No)	3 (Biom.)	1 (No)	1 (No)	[O,I] »$_n$ otp
App	A4 (Softw.)	1 (Desktop)	1 (No)	3 (Biom.)	1 (No)	2 (Yes)	[O,I] »$_h$ otp
App	A4 (Softw.)	1 (Desktop)	1 (No)	3 (Biom.)	~~2 (Yes)~~	–	not possible*
App	A4 (Softw.)	1 (Desktop)	2 (scan)	1 (No)	1 (No)	1 (No)	opid»$_o$ [O] »$_n$ otp
App	A4 (Softw.)	1 (Desktop)	2 (scan)	1 (No)	1 (No)	2 (Yes)	opid»$_o$ [O] »$_h$ otp
App	A4 (Softw.)	1 (Desktop)	2 (scan)	1 (No)	2 (Yes)	1 (No)	opid»$_o$ [O] »$_n$ otp
App	A4 (Softw.)	1 (Desktop)	2 (scan)	1 (No)	2 (Yes)	2 (Yes)	opid»$_o$ [O] »$_h$ otp
App	A4 (Softw.)	1 (Desktop)	2 (scan)	2 (PIN)	1 (No)	1 (No)	opid»$_o$ [O,K] »$_n$ otp
App	A4 (Softw.)	1 (Desktop)	2 (scan)	2 (PIN)	1 (No)	2 (Yes)	opid»$_o$ [O,K] »$_h$ otp
App	A4 (Softw.)	1 (Desktop)	2 (scan)	2 (PIN)	2 (Yes)	1 (No)	opid»$_o$ [O,K] »$_n$ otp
App	A4 (Softw.)	1 (Desktop)	2 (scan)	2 (PIN)	2 (Yes)	2 (Yes)	opid»$_o$ [O,K] »$_h$ otp
App	A4 (Softw.)	1 (Desktop)	2 (scan)	3 (Biom.)	1 (No)	1 (No)	opid»$_o$ [O,I] »$_n$ otp
App	A4 (Softw.)	1 (Desktop)	2 (scan)	3 (Biom.)	1 (No)	2 (Yes)	opid»$_o$ [O,I] »$_h$ otp
App	A4 (Softw.)	1 (Desktop)	2 (scan)	3 (Biom.)	2 (Yes)	1 (No)	opid»$_o$ [O,I] »$_n$ otp
App	A4 (Softw.)	1 (Desktop)	2 (scan)	3 (Biom.)	2 (Yes)	2 (Yes)	opid»$_o$ [O,I] »$_h$ otp
App	A4 (Softw.)	1 (Desktop)	3 (direct)	–	–	–	not possible◇
App	A4 (Softw.)	1 (Desktop)	4 (manual)	1 (No)	1 (No)	1 (No)	opid»$_h$ [O] »$_n$ otp
App	A4 (Softw.)	1 (Desktop)	4 (manual)	1 (No)	1 (No)	2 (Yes)	opid»$_h$ [O] »$_h$ otp
App	A4 (Softw.)	1 (Desktop)	4 (manual)	1 (No)	2 (Yes)	2 (No)	opid»$_h$ [O] »$_n$ otp

App	A4 (Softw.)	1 (Desktop)	4 (manual)	1 (No)	2 (Yes)	2 (Yes)	opid»$_h$ [O]? [O] »$_h$ otp
App	A4 (Softw.)	1 (Desktop)	4 (manual)	2 (PIN)	1 (No)	1 (No)	opid»$_h$ [O,K] »$_n$ otp
App	A4 (Softw.)	1 (Desktop)	4 (manual)	2 (PIN)	1 (No)	2 (Yes)	opid»$_h$ [O,K] »$_h$ otp
App	A4 (Softw.)	1 (Desktop)	4 (manual)	2 (PIN)	2 (Yes)	1 (No)	opid»$_h$ [O,K]? »$_n$ otp
App	A4 (Softw.)	1 (Desktop)	4 (manual)	2 (PIN)	2 (Yes)	2 (Yes)	opid»$_h$ [O,K]? »$_h$ otp
App	A4 (Softw.)	1 (Desktop)	4 (manual)	3 (Biom.)	1 (No)	1 (No)	opid»$_h$ [O,I] »$_n$ otp
App	A4 (Softw.)	1 (Desktop)	4 (manual)	3 (Biom.)	1 (No)	2 (Yes)	opid»$_h$ [O,K] »$_h$ otp
App	A4 (Softw.)	1 (Desktop)	4 (manual)	3 (Biom.)	2 (Yes)	1 (No)	opid»$_h$ [O,I]? »$_n$ otp
App	A4 (Softw.)	1 (Desktop)	4 (manual)	3 (Biom.)	2 (Yes)	2 (Yes)	opid»$_h$ [O,I]? »$_h$ otp
App	A4 (Softw.)	2 (Smartph.)	1 (No)	1 (No)	1 (No)	1 (No)	[O] »$_i$ otp
App	A4 (Softw.)	2 (Smartph.)	1 (No)	1 (No)	1 (No)	2 (Yes)	[O] »$_h$ otp
App	A4 (Softw.)	2 (Smartph.)	1 (No)	1 (No)	~~2 (Yes)~~	–	not possible*
App	A4 (Softw.)	2 (Smartph.)	1 (No)	2 (PIN)	1 (No)	1 (No)	[O,K] »$_i$ otp
App	A4 (Softw.)	2 (Smartph.)	1 (No)	2 (PIN)	1 (No)	2 (Yes)	[O,K] »$_h$ otp
App	A4 (Softw.)	2 (Smartph.)	1 (No)	2 (PIN)	~~2 (Yes)~~	–	not possible*
App	A4 (Softw.)	2 (Smartph.)	1 (No)	3 (Biom.)	1 (No)	1 (No)	[O,I] »$_i$ otp
App	A4 (Softw.)	2 (Smartph.)	1 (No)	3 (Biom.)	1 (No)	2 (Yes)	[O,I] »$_h$ otp
App	A4 (Softw.)	2 (Smartph.)	1 (No)	3 (Biom.)	~~2 (Yes)~~	–	not possible*
App	A4 (Softw.)	2 (Smartph.)	2 (scan)	1 (No)	1 (No)	1 (No)	opid»$_o$ [O] »$_i$ otp
App	A4 (Softw.)	2 (Smartph.)	2 (scan)	1 (No)	1 (No)	2 (Yes)	opid»$_o$ [O] »$_h$ otp
App	A4 (Softw.)	2 (Smartph.)	2 (scan)	1 (No)	2 (Yes)	2 (No)	opid»$_o$ [O]? »$_i$ otp
App	A4 (Softw.)	2 (Smartph.)	2 (scan)	1 (No)	2 (Yes)	2 (Yes)	opid»$_o$ [O]? »$_h$ otp
App	A4 (Softw.)	2 (Smartph.)	2 (scan)	2 (PIN)	1 (No)	1 (No)	opid»$_o$ [O,K] »$_i$ otp
App	A4 (Softw.)	2 (Smartph.)	2 (scan)	2 (PIN)	1 (No)	2 (Yes)	opid»$_o$ [O,K] »$_h$ otp
App	A4 (Softw.)	2 (Smartph.)	2 (scan)	2 (PIN)	2 (Yes)	1 (No)	opid»$_o$ [O,K]? »$_i$ otp
App	A4 (Softw.)	2 (Smartph.)	2 (scan)	2 (PIN)	2 (Yes)	2 (Yes)	opid»$_o$ [O,K]? »$_h$ otp
App	A4 (Softw.)	2 (Smartph.)	2 (scan)	3 (Biom.)	1 (No)	1 (No)	opid»$_o$ [O,I] »$_i$ otp
App	A4 (Softw.)	2 (Smartph.)	2 (scan)	3 (Biom.)	1 (No)	2 (Yes)	opid»$_o$ [O,I] »$_h$ otp
App	A4 (Softw.)	2 (Smartph.)	2 (scan)	3 (Biom.)	2 (Yes)	1 (No)	opid»$_o$ [O,I]? »$_i$ otp
App	A4 (Softw.)	2 (Smartph.)	2 (scan)	3 (Biom.)	2 (Yes)	2 (Yes)	opid»$_o$ [O,I]? »$_h$ otp
App	A4 (Softw.)	2 (Smartph.)	3 (direct)	1 (No)	1 (No)	1 (No)	opid»$_i$ [O] »$_i$ otp
App	A4 (Softw.)	2 (Smartph.)	3 (direct)	1 (No)	1 (No)	1 (Yes)	opid»$_i$ [O] »$_h$ otp
App	A4 (Softw.)	2 (Smartph.)	3 (direct)	1 (No)	2 (Yes)	2 (No)	opid»$_i$ [O]? »$_i$ otp
App	A4 (Softw.)	2 (Smartph.)	3 (direct)	1 (No)	2 (Yes)	2 (Yes)	opid»$_i$ [O]? »$_h$ otp
App	A4 (Softw.)	2 (Smartph.)	3 (direct)	2 (PIN)	1 (No)	1 (No)	opid»$_i$ [O,K] »$_i$ otp
App	A4 (Softw.)	2 (Smartph.)	3 (direct)	2 (PIN)	1 (No)	2 (Yes)	opid»$_i$ [O,K] »$_h$ otp
App	A4 (Softw.)	2 (Smartph.)	3 (direct)	2 (PIN)	2 (Yes)	1 (No)	opid»$_i$ [O,K]? »$_i$ otp
App	A4 (Softw.)	2 (Smartph.)	3 (direct)	2 (PIN)	2 (Yes)	2 (Yes)	opid»$_i$ [O,K]? »$_h$ otp
App	A4 (Softw.)	2 (Smartph.)	3 (direct)	3 (Biom.)	1 (No)	1 (No)	opid»$_i$ [O,I] »$_i$ otp
App	A4 (Softw.)	2 (Smartph.)	3 (direct)	3 (Biom.)	1 (No)	2 (Yes)	opid»$_i$ [O,I] »$_h$ otp
App	A4 (Softw.)	2 (Smartph.)	3 (direct)	3 (Biom.)	2 (Yes)	1 (No)	opid»$_i$ [O,I]? »$_i$ otp

App	A4 (Softw.)	2 (Smartph.)	3 (direct)	3 (Biom.)	2 (Yes)	2 (Yes)	opid$\gg_i$ [O,I] $\gg_h$ otp
App	A4 (Softw.)	2 (Smartph.)	4 (manual)	1 (No)	1 (No)	1 (No)	opid$\gg_h$ [O] $\gg_i$ otp
App	A4 (Softw.)	2 (Smartph.)	4 (manual)	1 (No)	1 (No)	2 (Yes)	opid$\gg_h$ [O] $\gg_h$ otp
App	A4 (Softw.)	2 (Smartph.)	4 (manual)	1 (No)	2 (Yes)	2 (No)	opid$\gg_h$ [O] $\gg_i$ otp
App	A4 (Softw.)	2 (Smartph.)	4 (manual)	1 (No)	2 (Yes)	2 (Yes)	opid$\gg_h$ [O] $\gg_h$ otp
App	A4 (Softw.)	2 (Smartph.)	4 (manual)	2 (PIN)	1 (No)	1 (No)	opid$\gg_h$ [O,K] $\gg_i$ otp
App	A4 (Softw.)	2 (Smartph.)	4 (manual)	2 (PIN)	1 (No)	2 (Yes)	opid$\gg_h$ [O,K] $\gg_h$ otp
App	A4 (Softw.)	2 (Smartph.)	4 (manual)	2 (PIN)	2 (Yes)	1 (No)	opid$\gg_h$ [O,K] $\gg_i$ otp
App	A4 (Softw.)	2 (Smartph.)	4 (manual)	2 (PIN)	2 (Yes)	2 (Yes)	opid$\gg_h$ [O,K] $\gg_h$ otp
App	A4 (Softw.)	2 (Smartph.)	4 (manual)	3 (Biom.)	1 (No)	1 (No)	opid$\gg_h$ [O,I] $\gg_i$ otp
App	A4 (Softw.)	2 (Smartph.)	4 (manual)	3 (Biom.)	1 (No)	2 (Yes)	opid$\gg_h$ [O,I] $\gg_h$ otp
App	A4 (Softw.)	2 (Smartph.)	4 (manual)	3 (Biom.)	2 (Yes)	1 (No)	opid$\gg_h$ [O,I] $\gg_i$ otp
App	A4 (Softw.)	2 (Smartph.)	4 (manual)	3 (Biom.)	2 (Yes)	2 (Yes)	opid$\gg_h$ [O,I] $\gg_h$ otp

Table 4: Table of possible sequences of forms after selecting "Out-of-band" option; ♣ we do not consider phone locking mechanisms as additional factors; $^{\nabla}$the output cannot be different than the code transmitted to the telephony server; *since the authenticator does not receive an input, it cannot show info on the ongoing operation.

FE	**FA**	**FO** Source	**F1** Input	**F2** Addit. Factor	**F3** Confirm.	**F4** Manual copy	**Authenticator**
Deskt.	A5 (OOB)	1 (SMS)	–	1 (No)	1 (No)	1 (No)	otp$\gg_m$ [O] $\gg_m$ otp
Deskt.	A5 (OOB)	1 (SMS)	–	1 (No)	1 (No)	2 (Yes)	otp$\gg_m$ [O] $\gg_h$ otp
Deskt.	A5 (OOB)	1 (SMS)	–	1 (No)	2 (Yes)	1 (No)	opid$\gg_m$ [O] $\gg_m$ otp
Brows.	A5 (OOB)	1 (SMS)	–	1 (No)	2 (Yes)	2 (Yes)	opid$\gg_m$ [O] $\gg_h$ otp
Brows.	A5 (OOB)	1 (SMS)	–	~~2 (PIN)~~	–	–	not possible♣
Brows.	A5 (OOB)	1 (SMS)	–	~~3 (Biom.)~~	–	–	not possible♣
Brows.	A5 (OOB)	2 (Call)	1 (No)	1 (No)	1 (No)	1 (No)	[O] $\gg_m$ otp
Brows.	A5 (OOB)	2 (Call)	1 (No)	1 (No)	1 (No)	~~2 (Yes)~~	not possible$^{\nabla}$
Brows.	A5 (OOB)	2 (Call)	1 (No)	1 (No)	~~2 (Yes)~~	–	not possible*
Brows.	A5 (OOB)	2 (Call)	1 (No)	~~2 (PIN)~~	–	–	not possible♣
Brows.	A5 (OOB)	2 (Call)	1 (No)	~~3 (Biom.)~~	–	–	not possible♣
Brows.	A5 (OOB)	2 (Call)	2 (Yes)	1 (No)	1 (No)	1 (No)	opid$\gg_h$ [O] $\gg_m$ otp
Brows.	A5 (OOB)	2 (Call)	2 (Yes)	1 (No)	1 (No)	~~2 (Yes)~~	not possible$^{\nabla}$
Brows.	A5 (OOB)	2 (Call)	2 (Yes)	1 (No)	2 (Yes)	1 (No)	opid$\gg_h$ [O] $\gg_m$ otp
Brows.	A5 (OOB)	2 (Call)	2 (Yes)	1 (No)	2 (Yes)	~~2 (Yes)~~	not possible$^{\nabla}$
Brows.	A5 (OOB)	2 (Call)	2 (Yes)	~~2 (PIN)~~	–	–	not possible♣
Brows.	A5 (OOB)	2 (Call)	2 (Yes)	~~3 (Biom.)~~	–	–	not possible♣
Brows.	A5 (OOB)	3 (Notif.)	–	1 (No)	1 (No)	1 (No)	opid$\gg_n$ [O] $\gg_n$ otp

Brows.	A5 (OOB)	3 (Notif.)	–	1 (No)	1 (No)	2 (Yes)	`opid`$\gg_n$📱[O] $\gg_h$ `otp`
Brows.	A5 (OOB)	3 (Notif.)	–	1 (No)	2 (Yes)	1 (No)	`opid`$\gg_n$📱?[O] $\gg_n$ `otp`
Brows.	A5 (OOB)	3 (Notif.)	–	1 (No)	2 (Yes)	2 (Yes)	`opid`$\gg_n$📱?[O] $\gg_h$ `otp`
Brows.	A5 (OOB)	3 (Notif.)	–	2 (PIN)	1 (No)	1 (No)	`opid`$\gg_n$📱[O,K] $\gg_n$ `otp`
Brows.	A5 (OOB)	3 (Notif.)	–	2 (PIN)	1 (No)	2 (Yes)	`opid`$\gg_n$📱[O,K] $\gg_h$ `otp`
Brows.	A5 (OOB)	3 (Notif.)	–	2 (PIN)	2 (Yes)	1 (No)	`opid`$\gg_n$📱?[O,K] $\gg_n$ `otp`
Brows.	A5 (OOB)	3 (Notif.)	–	2 (PIN)	2 (Yes)	2 (Yes)	`opid`$\gg_n$📱?[O,K] $\gg_h$ `otp`
Brows.	A5 (OOB)	3 (Notif.)	–	3 (Biom.)	1 (No)	1 (No)	`opid`$\gg_n$📱[O,I] $\gg_n$ `otp`
Brows.	A5 (OOB)	3 (Notif.)	–	3 (Biom.)	1 (No)	2 (Yes)	`opid`$\gg_n$📱[O,I] $\gg_h$ `otp`
Brows.	A5 (OOB)	3 (Notif.)	–	3 (Biom.)	2 (Yes)	1 (No)	`opid`$\gg_n$📱?[O,I] $\gg_n$ `otp`
Brows.	A5 (OOB)	3 (Notif.)	–	3 (Biom.)	2 (Yes)	2 (Yes)	`opid`$\gg_n$📱?[O,I] $\gg_h$ `otp`
App	A5 (OOB)	1 (SMS)	–	1 (No)	1 (No)	1 (No)	`otp`$\gg_m$ 🖷[O] $\gg_i$ `otp`
App	A5 (OOB)	1 (SMS)	–	1 (No)	1 (No)	2 (Yes)	`otp`$\gg_m$ 🖷[O] $\gg_h$ `otp`
App	A5 (OOB)	1 (SMS)	–	1 (No)	2 (Yes)	1 (No)	`opid`$\gg_m$ 🖷?[O] $\gg_i$ `otp`
App	A5 (OOB)	1 (SMS)	–	1 (No)	2 (Yes)	2 (Yes)	`opid`$\gg_m$ 🖷?[O] $\gg_h$ `otp`
App	A5 (OOB)	1 (SMS)	–	~~2 (PIN)~~	–	–	not possible
App	A5 (OOB)	1 (SMS)	–	~~3 (Biom.)~~	–	–	not possible
App	A5 (OOB)	2 (Call)	1 (No)	1 (No)	1 (No)	1 (No)	🖷[O] $\gg_m$ `otp`
App	A5 (OOB)	2 (Call)	1 (No)	1 (No)	1 (No)	~~2 (Yes)~~	not possible$^{\nabla}$
App	A5 (OOB)	2 (Call)	1 (No)	1 (No)	~~2 (Yes)~~	–	not possible*
App	A5 (OOB)	2 (Call)	1 (No)	~~2 (PIN)~~	–	–	not possible$^{\clubsuit}$
App	A5 (OOB)	2 (Call)	1 (No)	~~3 (Biom.)~~	–	–	not possible$^{\clubsuit}$
App	A5 (OOB)	2 (Call)	2 (Yes)	1 (No)	1 (No)	1 (No)	`opid`$\gg_h$ 🖷[O] $\gg_m$ `otp`
App	A5 (OOB)	2 (Call)	2 (Yes)	1 (No)	1 (No)	~~2 (Yes)~~	not possible$^{\nabla}$
App	A5 (OOB)	2 (Call)	2 (Yes)	1 (No)	2 (Yes)	1 (No)	`opid`$\gg_h$ 🖷?[O] $\gg_m$ `otp`
App	A5 (OOB)	2 (Call)	2 (Yes)	1 (No)	2 (Yes)	~~2 (Yes)~~	not possible$^{\nabla}$
App	A5 (OOB)	2 (Call)	2 (Yes)	~~2 (PIN)~~	–	–	not possible$^{\clubsuit}$
App	A5 (OOB)	2 (Call)	2 (Yes)	~~3 (Biom.)~~	–	–	not possible$^{\clubsuit}$
App	A5 (OOB)	3 (Notif.)	–	1 (No)	1 (No)	1 (No)	`opid`$\gg_n$📱[O] $\gg_i$ `otp`
App	A5 (OOB)	3 (Notif.)	–	1 (No)	1 (No)	2 (Yes)	`opid`$\gg_n$📱[O] $\gg_h$ `otp`
App	A5 (OOB)	3 (Notif.)	–	1 (No)	2 (Yes)	1 (No)	`opid`$\gg_n$📱?[O] $\gg_i$ `otp`
App	A5 (OOB)	3 (Notif.)	–	1 (No)	2 (Yes)	2 (Yes)	`opid`$\gg_n$📱?[O] $\gg_h$ `otp`
App	A5 (OOB)	3 (Notif.)	–	2 (PIN)	1 (No)	1 (No)	`opid`$\gg_n$📱[O,K] $\gg_i$ `otp`
App	A5 (OOB)	3 (Notif.)	–	2 (PIN)	1 (No)	2 (Yes)	`opid`$\gg_n$📱[O,K] $\gg_h$ `otp`
App	A5 (OOB)	3 (Notif.)	–	2 (PIN)	2 (Yes)	1 (No)	`opid`$\gg_n$📱?[O,K] $\gg_i$ `otp`
App	A5 (OOB)	3 (Notif.)	–	2 (PIN)	2 (Yes)	2 (Yes)	`opid`$\gg_n$📱?[O,K] $\gg_h$ `otp`
App	A5 (OOB)	3 (Notif.)	–	3 (Biom.)	1 (No)	1 (No)	`opid`$\gg_n$📱[O,I] $\gg_i$ `otp`
App	A5 (OOB)	3 (Notif.)	–	3 (Biom.)	1 (No)	2 (Yes)	`opid`$\gg_n$📱[O,I] $\gg_h$ `otp`
App	A5 (OOB)	3 (Notif.)	–	3 (Biom.)	2 (Yes)	1 (No)	`opid`$\gg_n$📱?[O,I] $\gg_i$ `otp`
App	A5 (OOB)	3 (Notif.)	–	3 (Biom.)	2 (Yes)	2 (Yes)	`opid`$\gg_n$📱?[O,I] $\gg_h$ `otp`

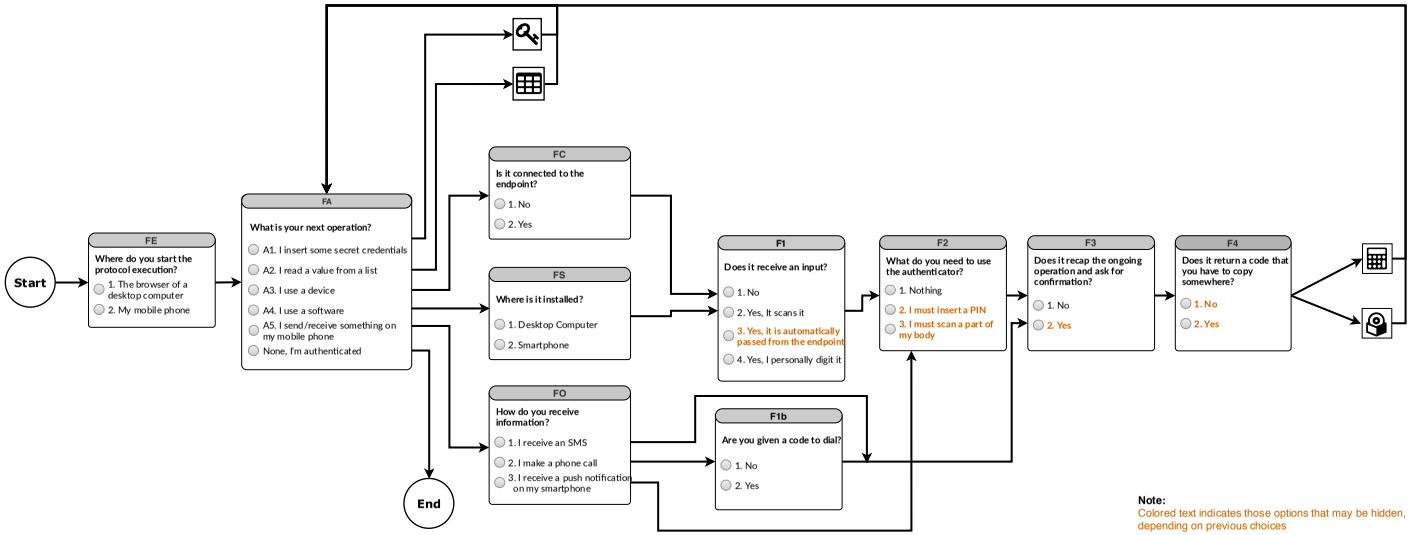


Figure 1: Available form sequences in MuFASA

MFA implementations per bank

In this section, we detail the MFA implementations adopted by the considered banks. Table 5 introduces the notation. The MFA protocols employed by the banks are presented in Table 6. An overview on the MFA implementations adopted by each bank is given in Tables 7 and 8, reporting the enrollment procedure, the employed authenticators and the respective binding procedures, the employed MFA protocols (both for Internet and Mobile Payments) and the adoption of exemptions.

Table 5: Notation for authenticators, data I/O, exemptions, enrollment & binding.

AUTHENTICATORS		
Authenticator type	**Data items**	**Data channels**
Memorized secret	`opid` Operation identifier	$\gg_h$ Manual copy
Look-up secret	`otp` One-time password	$\gg_i$ Inter-process communication
Device authenticator		$\gg_o$ Optical code scan
Software authenticator		$\gg_m$ Mobile telephony network
Out-of-band device auth.		$\gg_n$ Network packet
Out-of-band software auth.		

EXEMPTIONS		
✖ No exemptions	✔ Personal data visualization	✔✔ Low risk payments

ENROLLMENT AND BINDING	
The user goes to a local branch	The user establishes a remote session
† The user runs a MFA protocol (†binding only)	**E**† The user operates during the enrollment

Table 6: MFA protocols adopted by banks.

		Deutsche bank VR Bank Commerzbank	HSBC Barclays Lloyds bank	BNP Paribas Credit Agricole Societè Générale	Unicredit Intesa Sanpaolo Banco BPM	Banco Santander BBVA La Caixa	ING Rabobank ABN AMRO	Nordea Svenska Handelsb. SEB	ICBC CCB ABC	Chase Bank Of America Wells Fargo	UBS Credit Suisse Raiffeisen
		DE	UK	FR	IT	ES	NL	SW	CN	US	CH
ID	**MFA protocol**	**Internet Payments**									
IP-1	🔑;▦[O] »$_h$ otp				✓ ✓ ✓						✓
IP-2	🔑;▦[O,K] »$_h$ otp		✓ ✓		✓				✓		
IP-3	🔑;opid »$_i$ ▦?[O] »$_i$ otp								✓ ✓		✓
IP-4	🔑;🔑;opid »$_i$ ▦?[O] »$_i$ otp								✓		
IP-5	🔑;opid »$_i$ ▦?[O,K] »$_i$ otp						✓	✓ ✓ ✓			
IP-6	🔑;opid »$_o$ ▦?[O] »$_h$ otp	✓ ✓ ✓									✓
IP-7	🔑;opid »$_o$ ▦?[O,K] »$_h$ otp						✓				
IP-8	🔑;opid »$_h$ ▦[O,K] »$_h$ otp							✓ ✓ ✓			✓
IP-9	🔑;🔑;opid »$_h$ ▦[O,K] »$_h$ otp								✓		
IP-10	🔑;opid »$_h$ ▦?[O,K] »$_h$ otp						✓ ✓				
IP-11	🔑;▤	✓ ✓			✓	✓	✓	✓	✓ ✓		✓
IP-12	🔑;🔑;▤								✓		
IP-13	🔑;opid »$_i$ 🔒?[O,K] »$_i$ otp							✓ ✓			
IP-14	🔑;otp »$_m$ 🖩 »$_h$ otp			✓ ✓ ✓						✓ ✓ ✓	✓ ✓
IP-15	🔑;otp »$_m$ 🖩? »$_h$ otp	✓ ✓ ✓				✓	✓		✓ ✓		
IP-16	🔑;🔑;otp »$_m$ 🖩 »$_h$ otp					✓					
IP-17	🔑;🔑;otp »$_h$ 🖩 »$_m$ otp		✓								
IP-18	🔑;🔒[O] »$_h$ otp				✓						
IP-19	🔑;🔒[O,K] »$_h$ otp		✓ ✓								
IP-20	🔑;🔒[O,I] »$_h$ otp		✓ ✓								
IP-21	🔑;opid »$_o$ 🔒?[O] »$_h$ otp	✓ ✓									
IP-22	🔑;opid »$_o$ 🔒[O,K] »$_h$ otp										✓ ✓
IP-23	🔑;opid »$_o$ 🔒?[O,K] »$_h$ otp										✓
IP-24	🔑;opid »$_h$ 🔒?[O,K] »$_h$ otp				✓						
IP-25	🔑;opid »$_h$ 🔒?[O,I] »$_h$ otp				✓						
IP-26	🔑;opid »$_n$ 📱?[O] »$_o$ otp					✓					
IP-27	🔑;opid »$_n$ 📱?[O,K] »$_n$ otp			✓ ✓	✓ ✓		✓	✓ ✓ ✓			
IP-28	🔑;opid »$_n$ 📱?[O,I] »$_n$ otp				✓			✓ ✓ ✓			
IP-29	🔑;🔑;opid »$_n$ 📱[O,K] »$_n$ otp		✓								
IP-30	🔑;🔑;opid »$_n$ 📱[O,I] »$_n$ otp		✓								
IP-31	🔑;opid »$_n$ 📱?[O,K] »$_h$ otp	✓									
IP-32	🔑;opid »$_n$ 📱?[O,I] »$_h$ otp	✓									
ID	**MFA protocol**	**Mobile Payments**									
MP-1	🔑;▦[O] »$_h$ otp				✓ ✓						

Table 6: MFA protocols adopted by banks.

	Deutsche bank	VR Bank	Commerzbank	HSBC	Barclays	Lloyds bank	BNP Paribas	Credit Agricole	Societè Générale	Unicredit	Intesa Sanpaolo	Banco BPM	Banco Santander	BBVA	La Caixa	ING	Rabobank	ABN AMRO	Nordea	Svenska Handelsb.	SEB	ICBC	CCB	ABC	Chase	Bank Of America	Wells Fargo	UBS	Credit Suisse	Raiffeisen
	DE			UK			FR			IT			ES			NL			SW			CN			US			CH		
MP-2 🔑;▦[O,K]»$_h$ otp																						✓								
MP-3 🔑;opid»$_h$▦[O,K]»$_h$ otp																			✓	✓	✓							✓		
MP-4 🔑;🔑;opid»$_h$▦[O,K]»$_h$ otp																							✓							
MP-5 🔑;opid»$_h$▦?[O,K]»$_h$ otp																	✓													
MP-6 🔑;opid»$_i$▦?[O]»$_i$ otp																						✓	✓							
MP-7 🔑;🔑;opid»$_i$▦?[O]»$_i$ otp																							✓							
MP-8 🔑;opid»$_o$▦?[O]»$_h$ otp		✓	✓																											
MP-9 🔑;opid»$_o$▦?[O,K]»$_h$ otp																✓														
MP-10 🔑;▤	✓																		✓			✓	✓							
MP-11 🔑;🔑;▤																							✓							
MP-12 🔑;otp»$_m$🖬»$_i$ otp							✓																							✓
MP-13 🔑;otp»$_m$🖬?»$_i$ otp													✓										✓							
MP-14 🔑;🔑;otp»$_m$🖬»$_i$ otp													✓																	
MP-15 🔑;🔐[O,K]»$_i$ otp				✓																										
MP-16 🔑;🔐[O,I]»$_i$ otp				✓																										
MP-17 🔑;opid»$_i$🔐[O]»$_i$ otp						✓																								
MP-18 🔑;opid»$_i$🔐[O,K]»$_i$ otp																													✓	
MP-19 🔑;opid»$_i$🔐?[O]»$_i$ otp	✓		✓											✓		✓														
MP-20 🔑;opid»$_i$🔐?[O,K]»$_i$ otp		✓					✓		✓	✓	✓	✓							✓	✓	✓							✓		✓
MP-21 🔑;opid»$_i$🔐?[O,I]»$_i$ otp		✓								✓	✓								✓	✓	✓									
MP-22 🔐[I];opid»$_o$▦?[O]»$_h$ otp		✓	✓																											
MP-23 🔐[I];opid»$_h$▦?[O,K]»$_h$ otp																	✓													
MP-24 🔐[I];opid»$_o$▦?[O,K]»$_h$ otp																✓														
MP-25 🔐[I];▤	✓																													
MP-26 🔐[I];opid»$_i$🔐[O]»$_i$ otp						✓																								
MP-27 🔐[I];opid»$_i$🔐?[O]»$_i$ otp	✓		✓													✓														
MP-28 🔐[I];opid»$_i$🔐?[O,K]»$_i$ otp		✓																										✓		
MP-29 🔐[I];opid»$_i$🔐?[O,I]»$_i$ otp		✓																												

Table 7: MFA protocols adopted by EU banks.

Bank	C	Enr.	Authenticators	Binding	IP	MP	Ex
Deutsche Bank	DE	🏢	[I] opid»$_o$ [O] »$_h$ otp opid»$_o$ [O] »$_h$ otp	🌐,🌐,– **E**,🌐,🌐 🏢,🏢,🏢 🌐,🌐,🌐 🌐,🌐,🌐(📟)	IP-6, IP-11, IP-15, IP-21	MP-10, MP-19, MP-25, MP-27	✔
VR Bank	DE	🏢	[I] opid»$_o$ [O] »$_h$ otp otp»$_m$ »$_h$ otp opid»$_i$ [O,K] »$_i$ otp opid»$_n$ [O,K] »$_n$ otp opid»$_i$ [O,I] »$_i$ otp opid»$_n$ [O,I] »$_n$ otp	🌐, 🌐, 🌐 **E**,🌐,🔒 🏢, 🏢, 🌐 🌐,🌐,🌐 🌐,🌐,🌐 🌐,🌐,🌐 🌐,🌐,🌐	IP-6, IP-15, IP-31, IP-32	MP-8, MP-20, MP-21, MP-22, MP-28, MP-29	✔
Commerzbank	DE	🌐	[I] opid»$_o$ [O] »$_h$ otp otp»$_m$ »$_h$ otp opid»$_o$ [O] »$_h$ otp opid»$_i$ [O] »$_i$ otp	🌐, 🌐, – **E**,🌐,🌐 🌐,🌐,🌐 🌐,🌐,🌐 🌐,🌐,🌐(SMS) 🌐,🌐,🌐(SMS)	IP-6, IP-11, IP-15, IP-21	MP-8,MP-19, MP-22, MP-27	✔
HSBC	UK	🏢	[O,K] »$_h$ otp [O,K] »$_h$ otp [O,I] »$_h$ otp [O,K] »$_i$ otp [O,I] »$_i$ otp	**E**,🌐,🔒 🌐,🌐,🔒 🌐,🌐,🔒 🌐,🌐,🔒 🌐,🌐,🔒	IP-2, IP-19 IP-20	MP-15, MP-16	✔✔
Barclays	UK	🏢	 [O,K] »$_h$ otp [O,K] »$_h$ otp [O,I] »$_h$ otp	**E**, 🌐, 🌐 🌐,🌐,🔒 🌐, 🌐, 🔒 🌐, 🌐, 🔒	IP-2, IP-19, IP-20		✔✔
LLoyds Bank	UK	🏢	 otp»$_h$ 📟 »$_m$ otp opid»$_n$ [O,K] »$_n$ otp opid»$_n$ [O,I] »$_n$ otp [I] opid»$_i$ [O] »$_i$ otp	**E**, 🏢, 🏢 **E**,🏢,🏢 🌐,🌐,🌐 🌐,🌐,🌐 🌐,🌐,🌐 🌐,🌐,🌐	IP-17, IP-29, IP-30	MP-17, MP-26	✔✔
BNP Paribas	FR	🌐	otp»$_m$ 📟 »$_h$ otp opid»$_n$ [O,K] »$_n$ otp otp»$_i$ [O] »$_i$ otp	**E**,🌐,🌐 🌐,🌐,🌐(📟) 🌐,🌐,🌐(📟)	IP-14, IP-27	MP-12,MP-20	✔✔
Credit Agricole	FR	🌐	otp»$_m$ 📟 »$_h$ otp	**E**,🌐,🌐	IP-14		✔✔
Societe Generale	FR	🌐	otp»$_m$ 📟 »$_h$ otp opid»$_i$ [O,K] »$_i$ otp opid»$_n$ [O,K] »$_n$ otp	**E**,🌐,🌐 🌐,🌐,🌐(📟) 🌐,🌐,🌐(📟)	IP-14, IP-27	MP-20	✔

Table 7: MFA protocols adopted by EU banks.

Bank	C	Enr.	Authenticators	Binding	IP	MP	Ex
Unicredit	IT	📱	[O] »$_h$ otp ▦ opid»$_h$ [O,K] »$_h$ otp opid»$_h$ [O,I] »$_h$ otp opid»$_i$ [O,K] »$_i$ otp opid»$_i$ [O,K] »$_i$ otp	**E**, 🏢, 🏢 **E**, 🌐, 🌐 🌐, 🌐, 🔒 🌐, 🌐, 🔒 🌐, 🌐, 🔒 🌐, 🌐, 🔒	IP-1, IP-11, IP-24, IP-25	MP-20, MP-21	✔✔
Banca Intesa	IT	📱	[O] »$_h$ otp opid»$_n$ [O,K] »$_n$ otp opid»$_n$ [O,I] »$_n$ otp opid»$_i$ [O,K] »$_i$ otp opid»$_i$ [O,I] »$_i$ otp	**E**, 🏢, 🏢 🌐, 🌐, 🔒 🌐, 🌐, 🔒 🌐, 🌐, 🔒 🌐, 🌐, 🔒	IP-1, IP-27, IP-28	MP-1, MP-20, MP-21	✔✔
Banco BPM	IT	📱	[O] »$_h$ otp [O,K] »$_h$ otp [O] »$_h$ otp opid»$_n$ [O,K] »$_n$ otp opid»$_i$ [O,K] »$_i$ otp	**E**, 🏢, 🏢 🌐, 🏢, 🏢 🌐, 🌐, 🔒 🌐, 🌐, 🔒 🌐, 🌐, 🔒	IP-1, IP-2, IP-18, IP-27	MP-1, MP-20	✔
Banco Santander	ES	🌐	🔑 otp»$_m$ 📠 »$_h$ otp	**E**, 🌐, 🌐 **E**, 🌐, 🌐	IP-16	MP-14	✔✔
BBVA	ES	🌐	otp»$_m$ 📠 »$_h$ otp	**E**, 🌐, 🌐	IP-15	MP-13	✔
La Caixa	ES	🌐	▦ opid»$_n$ [O] »$_n$ otp opid»$_i$ [O] »$_i$ otp	**E**, 🏢, 🏢 🌐, 🌐, 🔒 🌐, 🌐, 🔒	IP-11, IP-26	MP-19	✔✔
ING Bank	NL	🌐	▦ otp»$_m$ 📠 »$_h$ otp opid»$_n$ [O,K] »$_n$ otp [I] opid»$_i$ [O] »$_i$ otp	**E**, 🌐, 🌐 **E**, 🌐, 🌐 🌐, 🌐, 🌐(📠) 🌐, 🌐, – 🌐, 🌐, 🌐(📠)	IP-11, IP-15, IP-27	MP-19, MP-27	✔
Rabobank	NL	🌐	opid»$_o$ [O,K] »$_h$ otp opid»$_h$ [O,K] »$_h$ otp [I]	**E**, 🏢, 🏢 – 🌐, 🌐, –	IP-7, IP-10	MP-9, MP-24	✔✔
ABN AMRO	NL	🌐	opid»$_i$ [O,K] »$_i$ otp opid»$_h$ [O,K] »$_h$ otp [I]	**E**, 🌐, 🌐 – 🌐, 🌐, –	IP-5, IP-10	MP-5, MP-5, MP-23, MP-23	✔✔
Nordea	SW	🌐	opid»$_h$ [O,K] »$_h$ otp opid»$_i$ [O,K] »$_i$ otp opid»$_i$ [O,K] »$_i$ otp opid»$_n$ [O,K] »$_n$ otp opid»$_n$ [O,I] »$_n$ otp opid»$_i$ [O,K] »$_i$ otp opid»$_i$ [O,I] »$_i$ otp	**E**, 🌐, 🌐 🌐, 🌐, 🌐 🌐, 🌐, 🔒 🌐, 🌐, 🔒 🌐, 🌐, 🔒 🌐, 🌐, 🔒 🌐, 🌐, 🔒	IP-5, IP-8, IP-13, IP-27, IP-28	MP-3, MP-20, MP-21	✔✔

Table 7: MFA protocols adopted by EU banks.

Bank	C	Enr.	Authenticators	Binding	IP	MP	Ex
Svenska Handelsbanken	SW		 opid»$_h$ [O,K] »$_h$ otp opid»$_i$ [O,K] »$_i$ otp opid»$_i$ [O,K] »$_i$ otp opid»$_n$ [O,K] »$_n$ otp opid»$_n$ [O,I] »$_n$ otp opid»$_i$  [O,K] »$_i$ otp opid»$_i$ [O,I] »$_i$ otp	**E**,, ,, ,, ,, ,, ,, ,, ,	IP-5, IP-8, IP-11, IP-13, IP-27, IP-28	MP-3, MP-10, MP-20, MP-21	✖
SEB	SW		opid»$_h$ [O,K] »$_h$ otp opid»$_i$ [O,K] »$_i$ otp opid»$_n$ [O,K] »$_n$ otp opid»$_n$ [O,I] »$_n$ otp opid»$_i$ [O,K] »$_i$ otp opid»$_i$ [O,I] »$_i$ otp	**E**,, ,, ,, ,, ,, ,	IP-5, IP-8, IP-27, IP-28	MP-3, MP-20, MP-21	✖

Table 8: MFA protocols adopted by non-EU banks.

Bank	C	Enr.	Authenticator	Binding	IP	MP	Ex
ICBC	CN		[O,K] »$_h$ otp opid»$_i$ [O] »$_i$ otp otp»$_m$ »$_h$ otp	, , **E**, , , , , ,	IP-2, IP-3, IP-11, IP-15	MP-2, MP-6, MP-10	✔
CCB	CN		opid»$_h$ [O,K] »$_h$ otp otp»$_m$ »$_h$ otp	, , **E**, , **E**, ,	IP-3, IP-11, IP-15	MP-6, MP-10, MP-13	✔
ABC	CN		opid»$_h$ [O,K] »$_h$ otp opid»$_i$ [O] »$_i$ otp 	**E**, , , , , , **E**, , –	IP-4, IP-9, IP-12	MP-4, MP-7, MP-11	✔
Chase	USA		otp»$_m$ »$_h$ otp	**E**, ,	IP-14		✔✔
Bank of America	USA		otp»$_m$  »$_h$ otp	**E**, ,	IP-14		✔✔
Wells Fargo	USA		otp»$_m$  »$_h$ otp	**E**, ,	IP-14		✔✔
UBS	CH		opid»$_h$ [O,K] »$_h$ otp opid»$_h$ [O,K] »$_h$ otp opid»$_i$ [O] »$_i$ otp [I] opid»$_o$ [O,K] »$_h$ otp opid»$_i$ [O,K] »$_i$ otp	**E**, , , , , , , , – , , , ,	IP-3, IP-8, IP-8, IP-22	MP-3, MP-3, MP-20, MP-28	✔✔

Table 8: MFA protocols adopted by non-EU banks.

Bank	C	Enr.	Authenticator	Binding	IP	MP	Ex
Credit Suisse	CH	🌐	[O] $\gg_h$ otp otp $\gg_m$ 📱 $\gg_h$ otp opid $\gg_o$ [O,K] $\gg_h$ otp opid $\gg_i$ [O,K] $\gg_i$ otp	**E**,🏢,🏢 **E**, 🌐, 🌐 🌐, 🌐, 🌐 🌐, 🌐, 🌐	IP-1, IP-14, IP-22	MP-18	✔✔
Raiffeisen	CH	🌐	▦ opid $\gg_o$ [O] $\gg_h$ otp otp $\gg_m$ 📱 $\gg_h$ otp opid $\gg_o$ [O,K] $\gg_h$ otp opid $\gg_i$ [O,K] $\gg_i$ otp	**E**, 🌐, 🌐 🌐, 🌐, 🔒 **E**, 🌐, 🌐 🔒, 🌐, 🔒 🔒, 🌐, 🔒	IP-6, IP-11, IP-14, IP-23	MP-12,MP-20	✖

Evaluation of resilience of MFA protocols against attacker models

In this section, we evaluate the observed MFA protocols in terms of complexity (with the three values related to *memory*, *manual operations* and *extra devices*) and resistance to the attacker models (namely *Device Thief*, *Authenticator Duplicator*, *Shoulder Surfer*, *Eavesdropping Software*, *Social Engineer*, *Man in the Browser* and *Man in the Mobile*) and their combinations. The results are presented in Tables 9 and 10 (for Internet and Mobile Payments, respectively).

Table 9: Analysis of the MFA protocols with respect to attackers models.

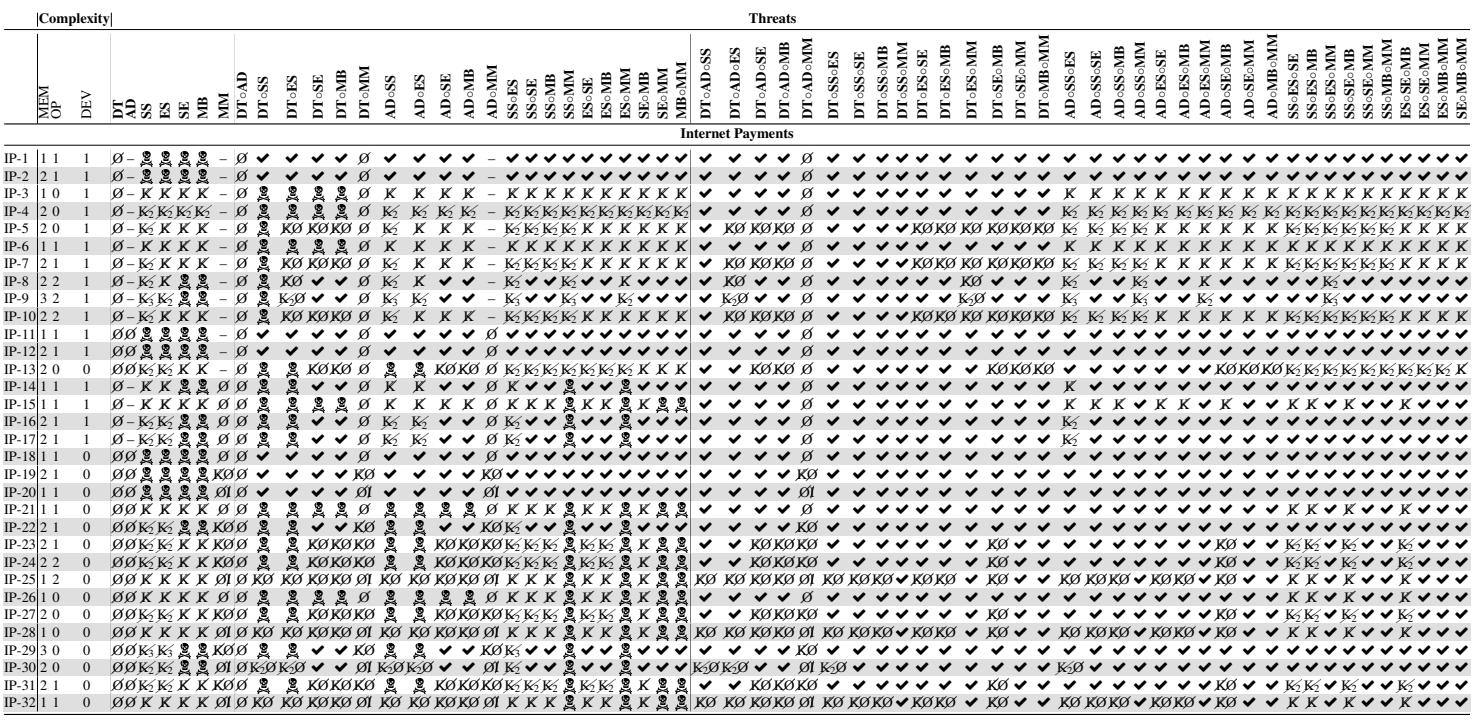
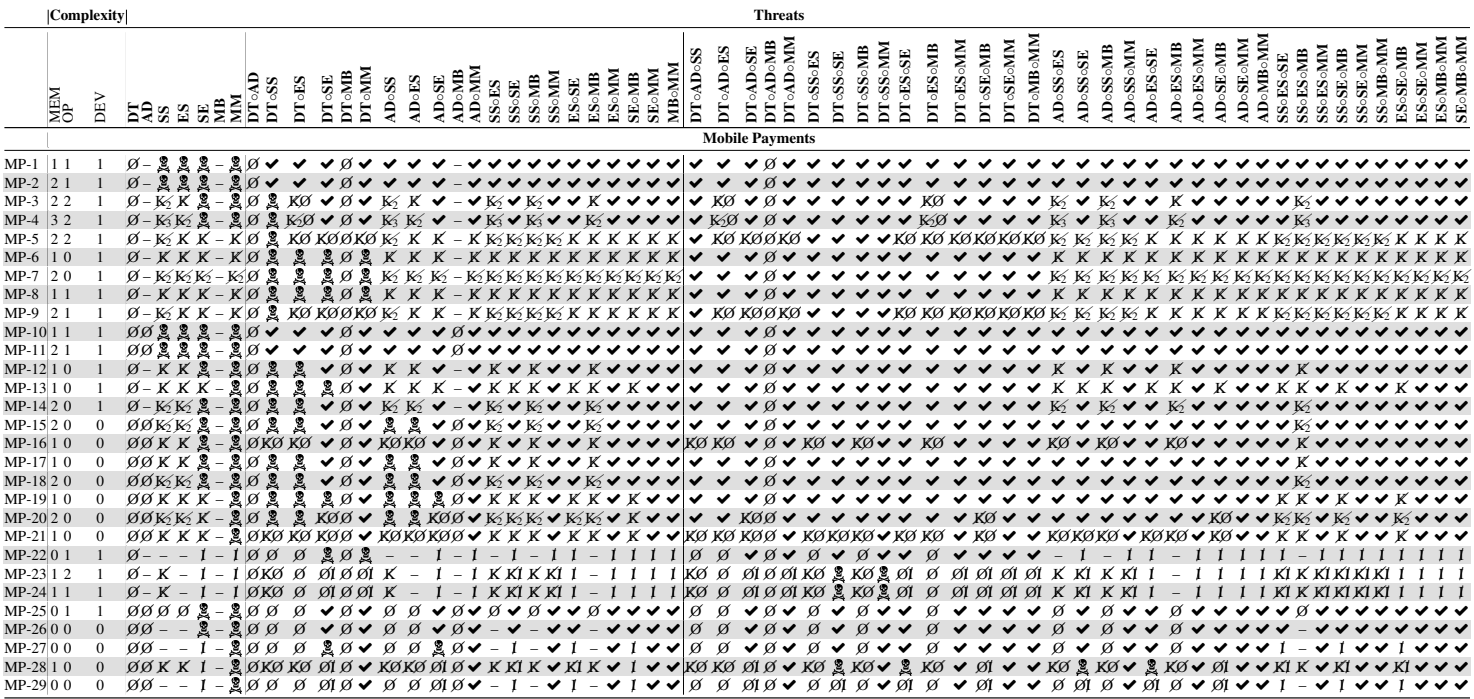


Table 10: Analysis of the MFA protocols with respect to attackers models.



Compliance with requirements and best practices

In this section, we present the compliance of EU and non-EU banks with the requirements extracted from the European regulations [Eur15, Eur17] and the best practices extracted from several guidelines [Cen16, PCI17, Gem15, NIS17, Pin09]. The results concerning EU banks are shown in Table 11, whereas those related to non-EU banks are shown in Table 12.

Table 11: Compliance with Requirements and Best Practices (EU banks).

Bank name	C	RL1	RL2	RL3	RL4	RL5	RL6	RL7	RL8	RL9	BP1	BP2	BP3	BP4	BP5	BP6	BP7	BP8
Deutsche Bank	DE	★	★	★	⯪	⯪	⯪	★	⯪	⯪	☆	⯪	★	⯪	★	⯪	⯪	★
VR Bank	DE	★	★	★	⯪	★	★	★	⯪	⯪	☆	⯪	★	⯪	★	⯪	⯪	★
Commerzbank	DE	★	★	★	⯪	⯪	⯪	⯪	☆	☆	☆	⯪	★	⯪	⯪	☆	⯪	★
HSBC	UK	★	★	★	★	☆	☆	★	★	★	★	⯪	★	★	★	★	⯪	★
Barclays	UK	★	★	★	★	☆	☆	★	⯪	⯪	★	⯪	★	★	★	⯪	☆	★
LLoyds	UK	★	★	★	⯪	★	⯪	★	⯪	⯪	★	★	★	⯪	★	⯪	☆	★
BNP Paribas	FR	★	★	★	★	★	⯪	⯪	☆	☆	★	⯪	★	⯪	⯪	☆	☆	★
Credit Agricole	FR	☆	★	★	★	★	☆	⯪	☆	☆	–	⯪	★	☆	⯪	☆	☆	☆
Sociètè Generale	FR	★	★	★	★	★	⯪	⯪	☆	☆	★	⯪	★	⯪	⯪	☆	☆	★
Unicredit	IT	★	★	★	★	⯪	⯪	★	⯪	⯪	★	⯪	★	★	★	⯪	★	★
Banca Intesa	IT	★	★	★	★	⯪	⯪	★	★	★	★	⯪	★	★	★	★	⯪	★
Banco BPM	IT	★	★	★	★	⯪	⯪	★	★	★	★	⯪	★	★	★	★	⯪	★
Banco Santander	ES	★	★	★	★	★	☆	⯪	☆	☆	–	⯪	★	☆	⯪	☆	☆	☆
BBVA	ES	★	★	★	★	★	★	⯪	☆	☆	–	⯪	★	☆	⯪	☆	☆	☆
La Caixa	ES	★	★	★	★	⯪	⯪	⯪	★	★	☆	⯪	★	★	⯪	★	⯪	★
ING Bank	NL	☆	★	★	⯪	⯪	⯪	⯪	☆	☆	★	⯪	★	⯪	⯪	☆	⯪	★
Rabobank	NL	☆	★	★	★	★	★	⯪	★	★	–	⯪	★	★	⯪	★	⯪	☆
ABN AMRO	NL	★	★	★	★	★	★	⯪	☆	☆	–	⯪	★	★	⯪	☆	⯪	☆
Nordea	SW	☆	★	★	★	★	⯪	⯪	⯪	⯪	☆	⯪	★	★	⯪	⯪	★	★
Svenska Handelsb.	SW	☆	★	★	★	⯪	⯪	★	⯪	⯪	☆	⯪	☆	★	★	⯪	★	★
SEB	SW	☆	★	★	★	★	⯪	⯪	⯪	⯪	☆	⯪	☆	★	⯪	⯪	★	★

Legend: Requirements/best practices can be ★ fulfilled, ⯪ partially violated or ☆ violated.

Table 12: Compliance with Requirements and Best Practices (non-EU banks).

Bank name	C	RL1	RL2	RL3	RL4	RL5	RL6	RL7	RL8	RL9	BP1	BP2	BP3	BP4	BP5	BP6	BP7	BP8
ICBC	CN	☆	★	★	★	⯪	⯪	★	★	★	–	☆	★	⯪	★	★	⯪	★
CCB	CN	☆	★	★	★	⯪	⯪	★	★	★	–	☆	★	⯪	★	★	⯪	★
ABC	CN	☆	★	★	★	⯪	⯪	★	★	★	–	☆	★	★	★	★	⯪	★
Chase	US	★	⯪	⯪	⯪	★	☆	⯪	☆	☆	–	★	★	☆	⯪	☆	☆	☆
Bank Of America	US	★	⯪	⯪	⯪	★	☆	⯪	☆	☆	–	⯪	★	☆	⯪	☆	☆	☆
Wells Fargo	US	★	⯪	⯪	⯪	★	☆	⯪	☆	☆	–	★	★	☆	⯪	☆	☆	☆
UBS	CH	★	★	★	⯪	★	⯪	⯪	★	★	☆	⯪	★	★	⯪	★	⯪	★
Credit Suisse	CH	★	★	★	★	⯪	⯪	⯪	⯪	⯪	☆	⯪	★	⯪	⯪	⯪	⯪	★
Raiffeisen	CH	★	★	★	★	⯪	⯪	⯪	⯪	⯪	☆	⯪	☆	⯪	⯪	⯪	⯪	★

Legend: Requirements/best practices can be ★ fulfilled, ⯪ partially violated or ☆ violated.